
THE HIVE
Will Du@Toronto

What Have We Learn So Far?

- What are two component for Apache Hadoop?
- What is Distributed Cache?
- What is Hadoop SE?
- What is Map Side Join?
- What is different between block and split?

What do we cover today?

- Hive overview and highlight
- Hive architecture
- Hive interface
- Hive data type and structure
- Hive database and tables
- Hive partitions
- Hive buckets
- Exercise in Class

Apache Hive Overview

- Data Warehousing Solution built on top of Hadoop
- Provides SQL-like query language named HiveQL - HQL, minimal learning curve
- Early Hive development work started at Facebook in 2007
- Today Hive is an top Apache project under Hadoop at hive.apache.org

Milestone of Hive

AUG 2007 – Journey Start

MAY 2013 – Hive 0.11.0 as Stinger Phase 1 - ORC, HiveSever2

OCT 2013 – Hive 0.12.0 as Stinger Phase 2 - ORC improvement

APR 2014 – Hive 0.13.0 as Stinger Phase 3 - vectorized query engine and Tez

NOV 2014 – Hive 0.14.0 as Stinger.next Phase 1- Cost-based optimizer

FEB 2015 – Hive 1.0.0

MAR 2015 – Hive 1.1.0

MAY 2015 – Hive 1.2.0

FEB 2016 – Hive 2.0.0 (add HPLSQL, LLAP)

Most BI and ETL tools leverage Hive as connector

The Stinger Initiative - Making Apache Hive **100** Times Faster

Highlight of Hive

- 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
- Hive supports running on different computing frameworks
- Hive supports ad hoc querying data on HDFS and HBase
- Hive supports user-defined functions, scripts, and customized I/O format to extend functionality
- Matured JDBC and ODBC drivers allow many applications to pull Hive data for seamless reporting
- Hive has a well-defined architecture for metadata management, authentication, optimizations
- There is a big community of practitioners and developers working on and using Hive

Hive vs. MapReduce – Word Count Again

```java
package org.apache.hadoop.examples;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);
private Text word = new Text();

public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
  StringTokenizer itr = new StringTokenizer(value.toString());
  while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    context.write(word, one);
    }
  }
}

public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,
                   Context context
                   ) throws IOException, InterruptedException {
  int sum = 0;
  for (IntWritable val : values) {
    sum += val.get();
  }
  result.set(sum);
  context.write(key, result);
  }
}

public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
if (otherArgs.length != 2) {
  System.err.println("Usage: wordcount <in> <out>");
  System.exit(2);
  }
Job job = new Job(conf, "word count");
job.setJarByClass(WordCount.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

67 Lines

Vs.

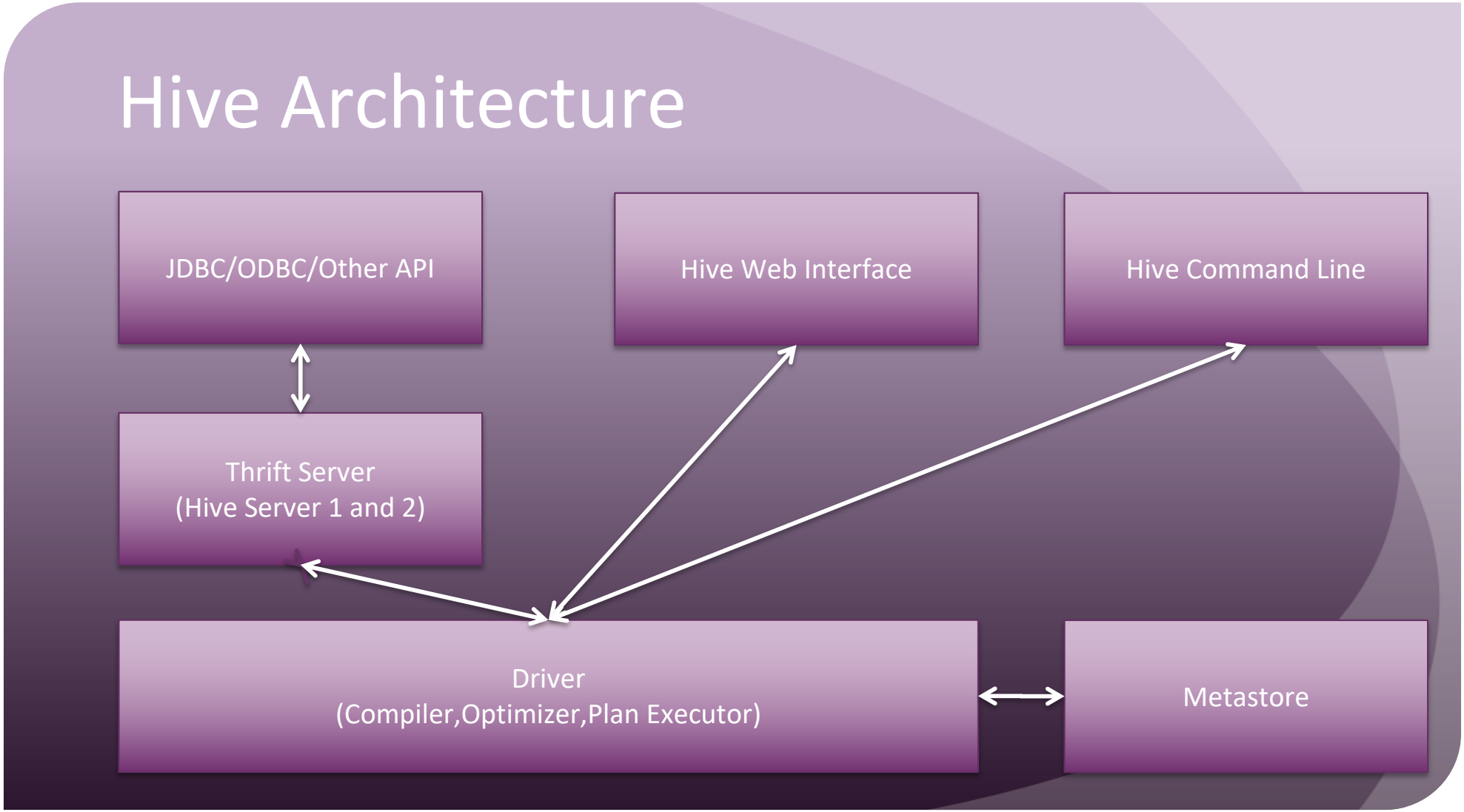
```
--Define metadata for the source
CREATE EXTERNAL TABLE lines(line STRING);
LOAD DATA INPATH 'book' OVERWRITE INTO TABLE lines;

-- word count
SELECT word, count(*) as world_count
FROM lines
LATERAL VIEW explode(split(text,' ')) t1 as word
GROUP BY word;
```

Hive Metadata

- To support features like schema(s) and data partitioning Hive keeps its metadata in a Relational Database
- By default, Hive is Packaged with Derby, a lightweight embedded SQL DB
  - ✧ Default Derby based is good for evaluation an testing
  - ✧ Schema is not shared between users as each user has their own instance of embedded Derby
  - ✧ Stored in metastore_db directory which resides in the directory that hive was started from
- Can easily switch another SQL installation such as MySQL
- HCatalog as part of Hive exposures Hive metadata to other ecosystem

Hive Architecture



Hive Interface – Command Line

- CML and Beeline
- Command Mode

Purpose	HiveServer2 Beeline	HiveServer1 CLI
Server connection	`beeline -u <jdbcurl> -n <username> -p <password>`	`hive -h <hostname> -p <port>`
Help	`beeline -h or beeline --help`	`hive -H`
Run query	`beeline -e <query in quotes>` `beeline -f <query file name>`	`hive -e <query in quotes>` `hive -f <query file name>`
Define variable	`beeline --hivevar key=value.` This is available after Hive 0.13.0.	`hive --hivevar key=value`

Hive Interface – Command Line Cont.

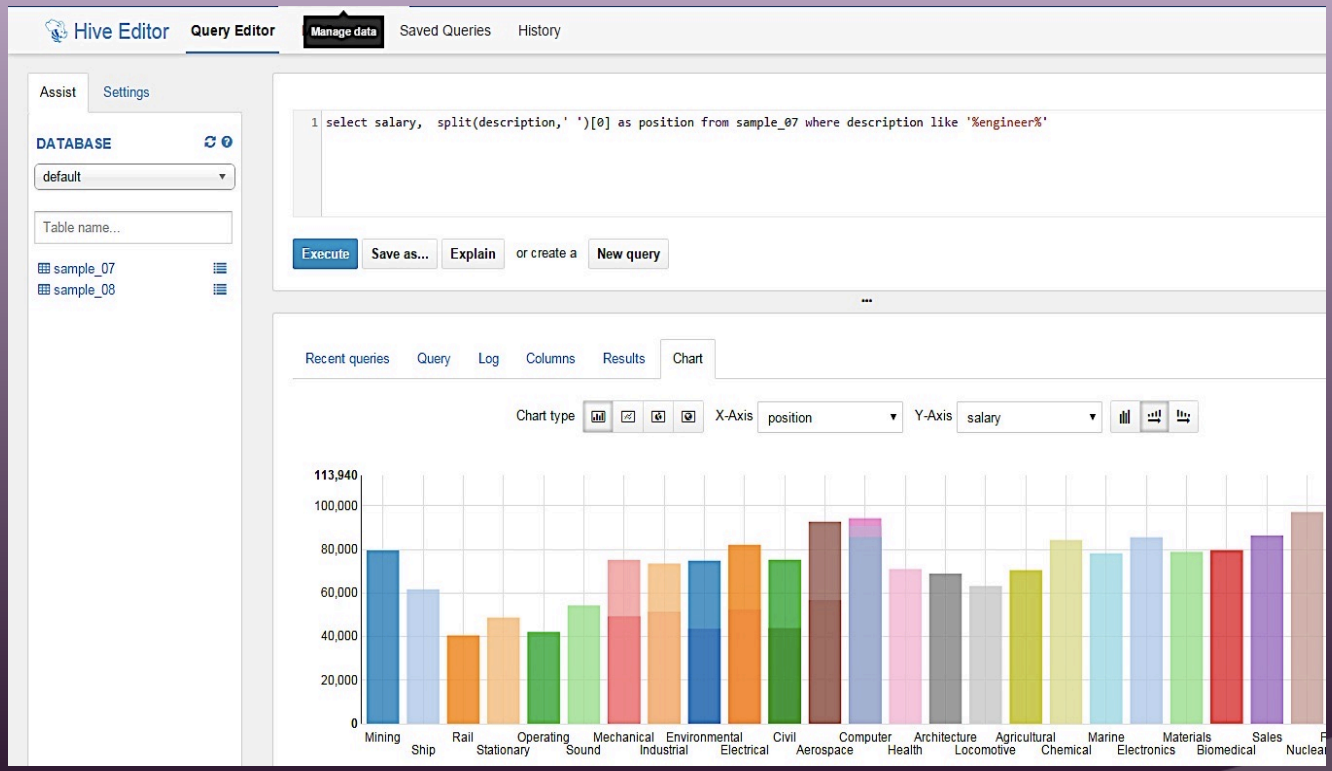
- Interactive Mode

Purpose	HiveServer2 Beeline	HiveServer1 CLI
Enter mode	`beeline`	`hive`
Connect	`!connect <jdbcurl>`	`n/a`
List tables	`!table`	`show tables;`
List columns	`!column <table_name>`	`desc <table_name>;`
Run query	`<HQL query>;`	`<HQL query>;`
Save result set	`!record <file_name>` `!record`	`N/A`
Run shell CMD	`!sh ls` This is available since Hive 0.14.0.	`!ls;`
Run dfs CMD	`dfs -ls`	`dfs -ls;`
Run file of SQL	`!run <file_name>`	`source <file_name>;`
Check Hive version	`!dbinfo`	`!hive --version;`
Quit mode	`!quit`	`quit;`

jdbc:hive2://ubuntu:11000/db2;user=foo;password=barc

Hive Interface – Others

Hive Web Interface|Hue|JDBC/ODBC|IDE



Demo 01 Quick Overview of Hue

http://demo.gethue.com/

Time: 10 min

Data Type – Primitive Type

Type	Example	Type	Example
TINYINT	10Y	SMALLINT	10S
INT	10	BIGINT	100L
FLOAT	1.342	DOUBLE	1.234
DECIMAL	3.14	BINARY	1010
BOOLEAN	TRUE	STRING	‘Book’ or “Book”
CHAR	‘YES’ or “YES”	VARCHAR	‘Book’ or “Book”
DATE	‘2013-01-31’	TIMESTAMP	‘2013-01-31 00:13:00.345’

Data Type – Complex Type

Type	Example	Define	Example
ARRAY	[‘Apple’,’Orange’,’Mongo’]	ARRAY<string>	a[0] = ‘Apple’
MAP	{‘A’:’Apple’,’O’:’Orange’}	MAP<string,string>	b[‘A’] = ‘Apple’
STRUCT	{‘Apple’, 2}	STRUCT<fruit:string,weight:int>	c.weight = 2

- ARRAY has same type for all the elements – equal to MAP uses sequence from 0 as keys
- MAP has same type of key value pairs
- STRUCT likes table/records

Hive Data Structure

Data Structure	Logical	Physical
Database	A collection of tables	Folder with files
Table	A collection of rows of data	Folder with files
Partition	Columns to split data	Folder
Buckets	Columns to distribute data	Files
Row	Line of records	Line in a file
Columns	Slice of records	Specified positions in each line
Views	Shortcut of rows of data	n/a
Index	Statistics of data	Folder with files

Hive Database

The database in describes a collection of tables that are used for a similar purpose or belong to the same group. If the database is not specified (use database_name), the default database is used. Hive creates a directory for each database at /user/hive/warehouse, which can be defined through hive.metastore.warehouse.dir property except default database.

- CREATE DATABASE IF NOT EXISTS myhivebook;
- SHOW DATABASES;
- DESCRIBE DATABASE default; //Provide more details than 'SHOW', such as comments, location
- DROP DATABASE IF EXISTS myhivebook CASCADE;
- ALTER DATABASE myhivebook SET OWNER user dayongd;

Hive Tables - Concept

External Tables

Data is kept in the HDFS path specified by LOCATION keywords. The data is not managed by Hive fully since drop the table (metadata) will not delete the data

Internal Tables/Managed Table

Data is kept in the default path, such as /user/hive/warehouse/employee. The data is fully managed by Hive since drop the table (metadata) will also delete the data

Hive Tables - DDL

```
1  Michael|Montreal,Toronto|Male,30|DB:80|Product:Developer^DLead
2  Will|Montreal|Male,35|Perl:85|Product:Lead,Test:Lead
3  Shelley|New York|Female,27|Python:80|Test:Lead,COE:Architect
4  Lucy|Vancouver|Female,57|Sales:89,HR:94|Sales:Lead
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS employee_external (
   name string,
   work_place ARRAY<string>,
   sex_age STRUCT<sex:string,age:int>,
   skills_score MAP<string,int>,
   depart_title MAP<STRING,ARRAY<STRING>>
)
COMMENT 'This is an external table'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE
LOCATION '/user/dayongd/employee';
```

IF NOT EXISTS is optional
TEMPORARY table is also supported

List of columns with data type
There is no constrain support

Table comment is optional

How to separate columns

How to separate MAP and Collections

File format

Data files path in HDFS (URI)

Side Notes - Delimiters

The default delimiters in Hive are as follows:

- Field delimiter: Can be used with *Ctrl + A* or ^A (Use \001 when creating the table)
- Collection item delimiter: Can be used with *Ctrl + B* or ^B (\002)
- Map key delimiter: Can be used with *Ctrl + C* or ^C (\003)

Issues

If delimiter is overridden during the table creation, it only works in flat structure JIRA Hive-365.

For nested types, the level of nesting determines the delimiter, for example

- ARRAY of ARRAY – Outer ARRAY is ^B, inner ARRAY is ^C
- MAP of ARRAY – Outer MAP is ^C, inner ARRAY is ^D

Exercise 01 Table creation and complex data type

Time: 15 min

Exercise 01 Task

1. Download the working data from this [URL](https://raw.githubusercontent.com/datafibers/data_set/master/employee.txt) to /tmp folder
2. Upload the data to hdfs at /tmp/data/employee
3. Create database employee with proper comments, and following db property
   - 'creator'=your name'
   - 'date'='today's date YYYY-MM-DD'
4. Create an external table employee on this csv file on HDFS path above
5. Query the following data structure in the table
   - ARRAY
   - MAP
   - STRUCT

Hive Table – DDL – Create Table

- CTAS – CREATE TABLE ctas_employee as SELECT * FROM employee
- CTAS cannot create a partition, external, or bucket table
- CTAS with Common Table Expression (CTE)

```
CREATE TABLE cte_employee AS
WITH
r1 AS (SELECT name FROM r2 WHERE name = 'Michael'),
r2 AS  (SELECT name FROM employee WHERE sex_age.sex= 'Male'),
r3 AS  (SELECT name FROM employee  WHERE sex_age.sex= 'Female')
SELECT * FROM r1 UNION ALL SELECT * FROM r3;
```

- Create table like other table (fast), such as CREATE TABLE employee_like LIKE employee

Hive Table – Drop/Truncate/Alter Table

- DROP TABLE IF EXISTS employee [PERGE] statement removes metadata completely and move data to .Trash folder in the user home directory in HDFS if configured. With PERGE option (since Hive 0.14.0), the data is removed completely. When to drop an external table, the data is not removed. Such as DROP TABLE IF EXISTS employee;
- TRUNCATE TABLE employee statement removes all rows of data form an internal table.
- ALTER TABLE employee RENAME TO new_employee statement renames the table
- ALTER TABLE c_employee SET TBLPROPERTIES ('comment'='New name, comments') statement sets table property
- ALTER TABLE employee_internal SET SERDEPROPERTIES ('field.delim' = '$') statement set SerDe properties
- ALTER TABLE c_employee SET FILEFORMAT RCFILE statement sets file format.

Hive Table – Alter Table Columns

- ALTER TABLE employee_internal CHANGE name employee_name STRING [BEFORE|AFTER] sex_age statement can be used to change column name, position or type
- ALTER TABLE c_employee ADD COLUMNS (work string) statement add another column and type to the table at the end.
- ALTER TABLE c_employee REPLACE COLUMNS (name string) statement replace all columns in the table by specified column and type. After ALTER in this example, there is only one column in the table.

Note:

The ALTER TABLE statement will only modify the metadata of Hive, not actually data.

Exercise 02 Table creation further and Alter table

Time: 15 min

Exercise 02 Task

1. Download the working data from this URL to /tmp folder
2. Upload the data to hdfs at /tmp/data/employee
3. Create an internal table employee_internal
4. Load above data into this table
5. Create internal table ctas_employee by copying data from employee_internal
6. Create internal table cte_employee with male named Michael and all females
7. Try to create another empty table by copying schema of employee_internal - use two ways
8. Rename cte_employee to c_employee
9. Rename column name to employee_name in employee_internal table and also change column type
10. Add another column called work in c_employee table

Hive Partitions - Overview

- To increase performance Hive has the capability to partition data
  - ✧ The values of partitioned column divide a table into segments (folders)
  - ✧ Entire partitions can be ignored at query time
  - ✧ Similar to relational databases' but not as advanced
- Partitions have to be properly created by users. When inserting data must specify a partition
- There is no difference in schema between "partition" columns and regular columns
- At query time, Hive will automatically filter out partitions for better performance

Hive Partitions

```
CREATE TABLE employee_partitioned(
    name string,
    work_place ARRAY<string>,
    sex_age STRUCT<sex:string,age:int>,
    skills_score MAP<string,int>,
    depart_title MAP<STRING,ARRAY<STRING>>
)
PARTITIONED BY (Year INT, Month INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':';
```

Partition is **NOT** enabled automatically. We have to add/delete partition by ALTER TABLE statement

```
--Add multiple partitions
jdbc:hive2://> ALTER TABLE employee_partitioned ADD
. . . . . .> PARTITION (year=2014, month=11)
. . . . . .> PARTITION (year=2014, month=12);
No rows affected (0.248 seconds)

jdbc:hive2://> SHOW PARTITIONS employee_partitioned;
+---------------------+
|      partition      |
+---------------------+
| year=2014/month=11  |
| year=2014/month=12  |
+---------------------+
2 rows selected (0.108 seconds)

--Drop the partition
jdbc:hive2://> ALTER TABLE employee_partitioned
. . . . . .> DROP IF EXISTS PARTITION (year=2014, month=11);
jdbc:hive2://> SHOW PARTITIONS employee_partitioned;
+---------------------+
|      partition      |
+---------------------+
| year=2014/month=12  |
+---------------------+
1 row selected (0.107 seconds)
```

Hive Partitions – Dynamic Partitions

Hive also supports dynamically giving partition values. This is useful when data volume is large and we don't know what will be the partition values. To enable it, see line 1.

By default, the user must specify at least one static partition column. This is to avoid accidentally overwriting partitions. To disable this restriction, we can set the partition mode to nonstrict (see line 3) from the default strict mode.

```
1  SET hive.exec.dynamic.partition=true;
2
3  SET hive.exec.dynamic.partition.mode=nonstrict;
4
5  //example of dynamic partition
6  INSERT INTO TABLE employee_partitioned
7  PARTITION(year, month)
8  SELECT name, array('Toronto') as work_place,
9  named_struct("sex","Male","age",30) as sex_age,
10 map("Python",90) as skills_score,
11 map("R&D",array('Developer')) as depart_title,
12 year(start_date) as year,
13 month(start_date) as month
14 FROM employee_hr eh
15 WHERE eh.employee_id = 102;
```

Hive Buckets - Overview

- The bucket corresponds to the segments of files in the HDFS
- Mechanism to query and examine random samples of data and speed JOIN
- Break data into a set of buckets based on a hash function of a "bucket column”
- Hive doesn’t automatically enforce bucketing. User is required to specify the number of buckets by setting # of reducer **OR** set the enforced bucketing.

```
SET mapred.reduce.tasks = 2;
SET hive.enforce.bucketing = true;
```

- To define number of buckets, we should avoid having too much or too little data in each buckets. A better choice somewhere near two blocks of data. Use 2^N as the number of buckets

Hive Buckets - DDL

```
1  CREATE TABLE employee_id_buckets
2  (
3      name string,
4      employee_id int,
5      work_place ARRAY<string>,
6      sex_age STRUCT<sex:string,age:int>,
7      skills_score MAP<string,int>,
8      depart_title MAP<string,ARRAY<string >>
9  )
10 CLUSTERED BY (employee_id) INTO 2 BUCKETS
11 ROW FORMAT DELIMITED
12 FIELDS TERMINATED BY '|'
13 COLLECTION ITEMS TERMINATED BY ','
14 MAP KEYS TERMINATED BY ':';
15
16 INSERT OVERWRITE TABLE employee_id_buckets
17 SELECT * FROM employee_id;
18
19 --Verify the buckets in the HDFS
20 hdfs dfs -ls /user/hive/warehouse/employee_id_buckets
21
22 Found 2 items
23 /user/hive/warehouse/employee_id_buckets/000000_0
24 /user/hive/warehouse/employee_id_buckets/000001_0
25
```

- Use 'CLUSTERED BY' statements to define buckets (Line 10)
- To populate the data into the buckets, we have to use INSERT statement instead of LOAD statement (Line 16 – 17) since it does not verify the data against metadata definition (copy files to the folders only)
- Buckets are list of files segments (Line 23 and 24)

Exercise 03 Table partition and buckets

Time: 15 min

Exercise 03 Task

- Download the working data from this URL to /tmp folder
- Create empty partition table employee_partitioned using partition columns Year Int and Month Int
- Add two static partition to the table
- Load employee data into this partition table in one partition
- Create another external table employee_hr with data downloaded
- Load more data from employee_hr into this employee_partitioned with dynamic partition enabled
- Create another external table employee_id using data here
- Create a bucket table employee_id_buckets cluster by employee_id
- Load the bucket table from employee_id

Q&A

Thank You

Contact me at willddy@gmail.com

iT 21 LEARNING