

Partitioning in Hive

The partitioning in Hive means dividing the table into some parts based on the values of a particular column like date, course, city or country. The advantage of partitioning is that since the data is stored in slices, the query response time becomes faster.

As we know that Hadoop is used to handle the huge amount of data, it is always required to use the best approach to deal with it. The partitioning in Hive is the best example of it.

Let's assume we have a data of 10 million students studying in an institute. Now, we have to fetch the students of a particular course. If we use a traditional approach, we have to go through the entire data. This leads to performance degradation. In such a case, we can adopt the better approach i.e., partitioning in Hive and divide the data among the different datasets based on particular columns.

PARTITION:

- 1) Create a hive table
- 2)load the data in the hive table
- 3)create partition table
- 4)set hive.exec.dynamic.partition.mode=nonstrict;
- 5)load data into partition table from hive table.

EXAMPLES:

```
Hive> create table all_students(sno int, sname string, passout_year string)
```

```
>row format delimited fields terminated by ',';
```

OK

Time taken: 0.245 seconds

```
Hive> load data local inpath '/home/cloudera/training/hivedata/students.csv'  
into table all_students;
```

Loading data to table default.all_students

OK

Time taken: 0.423 seconds

```
Hive> select * from all_students;
```

OK

```
Hive> create table students_part(sno int, sname string) PARTITIONED BY  
(passout_year string);
```

Ok

Time taken:0.425 seconds

```
Hive> describe students_part;
```

```
Hive> set hive.exec.dynamic.partition.mode=nonstrict;
```

```
Hive> insert OVERWRITE TABLE students_part PARTITION(passout_year ) select  
sno,sname, passout_year from all_students;
```

OK

Time taken: 44.25 seconds

CREATE A NON-PARTITION TABLE:

Create table user_data_no_partition(sno int,user_name string,city string) ROW
FORMAT delimited fields terminated by ','

LINES TERMINATED BY '\n' STORED AS TEXTFILE;

Load data local inpath '/home/big/user_data.txt' into table user_data_no_partition;

Select * from user_data_no_partition;

PRACTICAL:

```
Hive> create table user_data_no_partition
```

```
>(sno int,user_name string,city string)
```

```
>ROW FORMAT delimited fields terminated by ',' LINES TERMINATED BY '\n'  
STORED AS TEXTFILE;
```

OK

TIME taken: 0.214 seconds

```
Hive> load data local inpath '/home/big/user_data.txt' into table user_data_no_partition;
```

Loading data to table default.user_data_no_partition

OK

Time taken: 0.918 seconds

```
Hive>select * from user_data_no_partition;
```

1, Gowtham, Chennai

2,Saravana,chennai

3, ram, delhi

4,alex, mumbai

5,Rahul ,delhi

6, arun ,goa

7, nila, chennai

8, nandini, chennai

9, anitha ,delhi

10, jaya, delhi

** For partitioned table,we cannot insert data by LOAD data command.

** We can only insert a data into portioned table with only INSERT INTO (or) INSERT OVERWRITE.

** By default, Hive will store all the tables in HDFS under

/user/hive/warehouse/ path only.

The partitioning in Hive can be executed in two ways -

1)Static partitioning

2)Dynamic Partitioning

Static Partitioning

In static or manual partitioning, it is required to pass the values of partitioned columns manually while loading the data into the table. Hence, the data file doesn't contain the partitioned columns.

STATIC PARTITION:

Example:1

Create table user_data

(sno int

User_name string)

Partitioned by(city string);

Insert into table user_data partition(city='chennai') select sno,user_name from user_data_no_partition where city='chennai';

Select * from user_data;

PRACTICAL

Hive> Create table user_data

>(sno intUser_name string)

>Partitioned by(city string);

OK

Time taken:0.259 seconds

```
Hive> Insert into table user_data partition(city='chennai') select sno,user_name  
from user_data_no_partition where city='chennai';
```

OK

Time taken:37.473 seconds

```
Hive> select * from user_data;
```

OK

1 gowtham chennai

2 saravana chennai

7 nila chennai

8 nandini chennai

Time taken: 0.591 seconds,fetched: 4 row(s)

```
Hive> show partitions user_data;
```

OK

City=Chennai

Time taken: 0.24 seconds, fetched: 1 row(s)

**** Insert into table user_data partition(city='chennai') select sno,user_name from
user_data_no_partition where city='chennai';**

**** We have to mention value in static partition . i.e.. city='chennai' (or) city='delhi'**

**** We cannot include 3rd column i.e.. partitioned column(city) in the select statement.**

WHERE and INSERT command must be same.

i.e.. (city='chennai'),where city='chennai;

.. (city='Delhi'),where city='Delhi;

If both are not same,we can get errors.

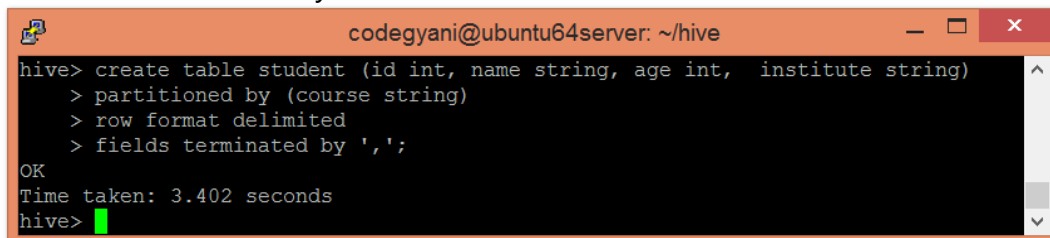
Example of Static Partitioning

- First, select the database in which we want to create a table.

hive> use test;

- Create the table and provide the partitioned columns by using the following command: -

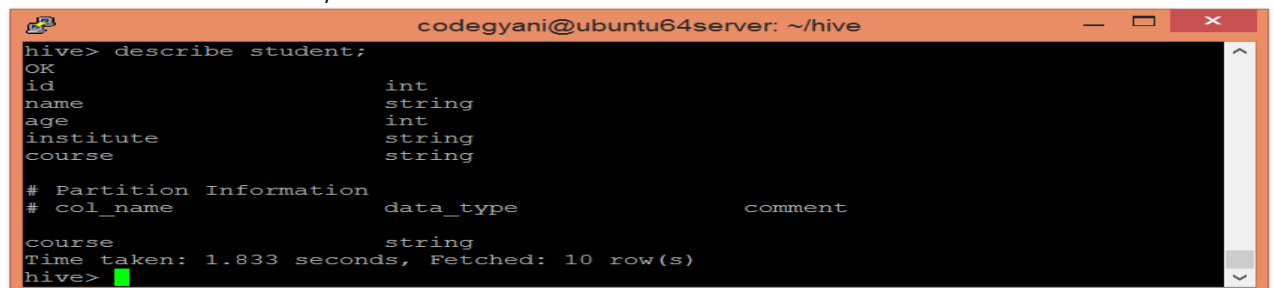
```
hive> create table student (id int, name string, age int, institute string)
partitioned by (course string)
row format delimited
fields terminated by ',';
```



```
codegyani@ubuntu64server: ~/hive
hive> create table student (id int, name string, age int, institute string)
> partitioned by (course string)
> row format delimited
> fields terminated by ',';
OK
Time taken: 3.402 seconds
hive>
```

- Let's retrieve the information associated with the table.

hive> describe student;



```
codegyani@ubuntu64server: ~/hive
hive> describe student;
OK
id                int
name              string
age              int
institute         string
course            string

# Partition Information
# col_name        data_type        comment
course            string
Time taken: 1.833 seconds, Fetched: 10 row(s)
hive>
```

- Load the data into the table and pass the values of partition columns with it by using the following command: -

```
hive> load data local inpath '/home/codegyani/hive/student_details1' into table
student
partition(course= "java");
```

```
codegyani@ubuntu64server: ~/hive
hive> load data local inpath '/home/codegyani/hive/student_details1' into table student
> partition(course= "java");
Loading data to table test.student partition (course=java)
Partition test.student{course=java} stats: [numFiles=1, numRows=0, totalSize=122, rawDataSize=0]
OK
Time taken: 8.057 seconds
hive>
```

Here, we are partitioning the students of an institute based on courses.

- Load the data of another file into the same table and pass the values of partition columns with it by using the following command: -

```
hive> load data local inpath '/home/codegyani/hive/student_details2' into table student
partition(course= "hadoop");
```

```
codegyani@ubuntu64server: ~/hive
hive> load data local inpath '/home/codegyani/hive/student_details2' into table student
> partition(course= "hadoop");
Loading data to table test.student partition (course=hadoop)
Partition test.student{course=hadoop} stats: [numFiles=1, numRows=0, totalSize=75, rawDataSize=0]
OK
Time taken: 2.402 seconds
hive>
```

In the following screenshot, we can see that the table student is divided into two categories.

Hadoop Overview Datanodes Snapshot Startup Progress Utilities							
Browse Directory							
<input type="text" value="/user/hive/warehouse/test.db/student"/>							<input type="button" value="Go"/>
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	codegyani	supergroup	0 B	8/1/2019, 5:39:27 PM	0	0 B	course=hadoop
drwxr-xr-x	codegyani	supergroup	0 B	8/1/2019, 5:37:13 PM	0	0 B	course=java

- Let's retrieve the entire data of the table by using the following command: -

```
hive> select * from student;
```

```
codegyani@ubuntu64server: ~/hive
hive> select * from student;
OK
6      "Chris" 22      javatpoint      hadoop
7      "Hariss" 21      javatpoint      hadoop
8      "Angelina"24      NULL      NULL      hadoop
1      "Gaurav" 24      javatpoint      java
2      "John" 22      javatpoint      java
3      "William" 21      javatpoint      java
4      "Steve" 24      javatpoint      java
5      "Roman" 23      javatpoint      java
Time taken: 4.85 seconds, Fetched: 8 row(s)
hive>
```

- Now, try to retrieve the data based on partitioned columns by using the following command: -

hive> select * from student where course="java";

```
codegyani@ubuntu64server: ~/hive
4      "Steve" 24      javatpoint      java
5      "Roman" 23      javatpoint      java
Time taken: 4.85 seconds, Fetched: 8 row(s)
hive> select * from student where course="java";
OK
1      "Gaurav" 24      javatpoint      java
2      "John" 22      javatpoint      java
3      "William" 21      javatpoint      java
4      "Steve" 24      javatpoint      java
5      "Roman" 23      javatpoint      java
Time taken: 4.166 seconds, Fetched: 5 row(s)
hive>
```

In this case, we are not examining the entire data. Hence, this approach improves query response time.

- Let's also retrieve the data of another partitioned dataset by using the following command: -

hive> select * from student where course="hadoop";

```
codegyani@ubuntu64server: ~/hive
hive> select * from student where course="hadoop";
OK
6      "Chris" 22      javatpoint      hadoop
7      "Hariss" 21      javatpoint      hadoop
8      "Angelina"24      NULL      NULL      hadoop
Time taken: 0.504 seconds, Fetched: 3 row(s)
hive>
```

Dynamic Partitioning

In dynamic partitioning, the values of partitioned columns exist within the table. So, it is not required to pass the values of partitioned columns manually.

Example1:

DYNAMIC PARTITION:

Create table user_data_dynamic

(sno int, user_name string)

Partitioned by (city string);

Set hive.exec.dynamic.partition.mode=nonstrict;

Insert into table user_data_dynamic partition(city) select sno,user_name,city from user_data_no_partition where city='chennai';

Select * from user_data_dynamic;

PRACTICAL:

```
Hive> Create table user_data_dynamic
```

```
>(sno int, user_name string)
```

```
>Partitioned by (city string);
```

```
Hive> Insert into table user_data_dynamic partition(city) select  
sno,user_name,city from user_data_no_partition where city='chennai';
```

OK

Time taken: 43.56 seconds

```
Hive> Set hive.exec.dynamic.partition.mode=nonstrict;
```

```
Hive>select * from user_data_dynamic;
```

OK

1 gowtham chennai

2 saravana chennai

7 nila chennai

8 nandini Chennai

Time taken: 0.4 seconds

** Insert into table user_data_dynamic partition(city) select sno,user_name,city from user_data_no_partition where city='chennai';

** We have mention only column name in insert statement,but column name is not necessary.

** We can 3rd column i.e.. partitioned column , because we are not giving the value for the column city.

** We can change WHERE condition.

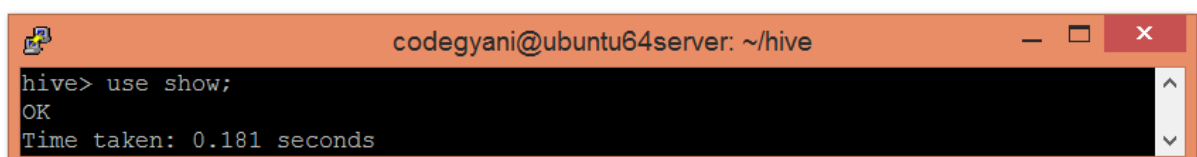
- By changing WHERE condition, we cannot get errors like static partition.

** In real time mostly we can use dynamic partition.

EXAMPLE2:

- First, select the database in which we want to create a table.

```
hive> use show;
```



```
codegyani@ubuntu64server: ~/hive
hive> use show;
OK
Time taken: 0.181 seconds
```

- Enable the dynamic partition by using the following commands: -

```
hive> set hive.exec.dynamic.partition=true;
```

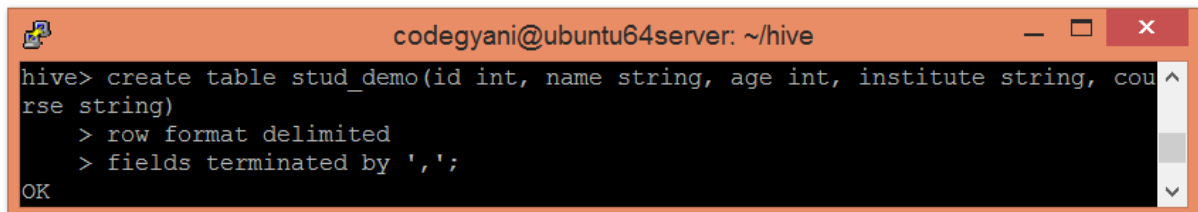
```
hive> set hive.exec.dynamic.partition.mode=nonstrict;
```

- Create a dummy table to store the data.

```
hive> create table stud_demo(id int, name string, age int, institute string, course string)
```

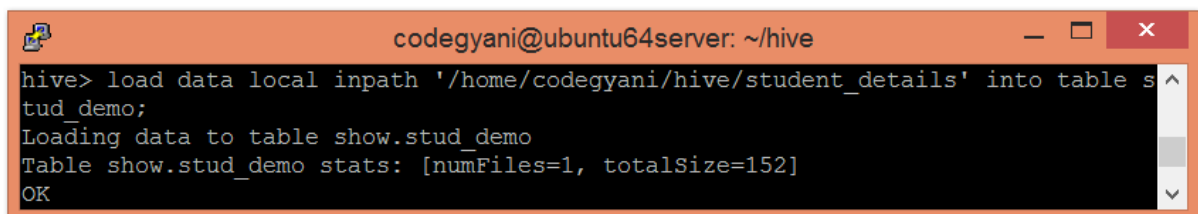
```
row format delimited
```

```
fields terminated by ',';
```

A terminal window titled 'codegyani@ubuntu64server: ~/hive' showing the execution of Hive commands. The commands are: 'hive> create table stud_demo(id int, name string, age int, institute string, course string)', '> row format delimited', and '> fields terminated by ',';'. The output is 'OK'.

- Now, load the data into the table.

```
hive> load data local inpath '/home/codegyani/hive/student_details' into table stud_demo;
```

A terminal window titled 'codegyani@ubuntu64server: ~/hive' showing the execution of Hive commands. The command is 'hive> load data local inpath '/home/codegyani/hive/student_details' into table stud_demo;'. The output is 'Loading data to table show.stud_demo' and 'Table show.stud_demo stats: [numFiles=1, totalSize=152]'. The final output is 'OK'.

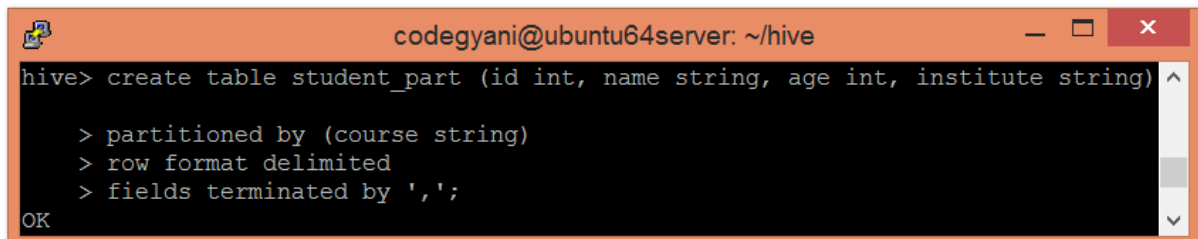
- Create a partition table by using the following command: -

```
hive> create table student_part (id int, name string, age int, institute string)
```

```
partitioned by (course string)
```

```
row format delimited
```

```
fields terminated by ',';
```

A terminal window titled 'codegyani@ubuntu64server: ~/hive' showing the execution of Hive commands. The commands are: 'hive> create table student_part (id int, name string, age int, institute string)', '> partitioned by (course string)', '> row format delimited', and '> fields terminated by ',';'. The output is 'OK'.

- Now, insert the data of dummy table into the partition table.

```
hive> insert into student_part
```

```
partition(course)
```

```
select id, name, age, institute, course
```

from stud_demo;

```
codegyani@ubuntu64server: ~/hive
hive> insert into student_part
> partition(course)
> select id, name, age, institute, course
> from stud_demo;
Query ID = codegyani_20190801062015_d7649030-f370-47a2-a86d-ff402d3e7de7
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1555046592674_0017, Tracking URL = http://ubuntu64server:8088
/proxy/application_1555046592674_0017/
Kill Command = /home/codegyani/hadoop-2.7.1/bin/hadoop job -kill job_155504659
2674_0017
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2019-08-01 06:21:50,531 Stage-1 map = 0%, reduce = 0%
2019-08-01 06:22:51,598 Stage-1 map = 0%, reduce = 0%
2019-08-01 06:23:06,456 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 10.03 s
ec
MapReduce Total cumulative CPU time: 10 seconds 30 msec
Ended Job = job_1555046592674_0017
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://192.168.56.123:8020/user/hive/warehouse/show.db/student_p
art/.hive-staging_hive_2019-08-01_06-20-15_112_3898950391086441478-1/-ext-10000
```

- In the following screenshot, we can see that the table student_part is divided into two categories.

Hadoop Overview Datanodes Snapshot Startup Progress Utilities							
Browse Directory							
/user/hive/warehouse/show.db/student_part							Go!
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	codegyani	supergroup	0 B	8/1/2019, 3:53:17 PM	0	0 B	course=__HIVE_DEFAULT_PARTITION__
drwxr-xr-x	codegyani	supergroup	0 B	8/1/2019, 3:53:15 PM	0	0 B	course=hadoop
drwxr-xr-x	codegyani	supergroup	0 B	8/1/2019, 3:53:16	0	0 B	course=java

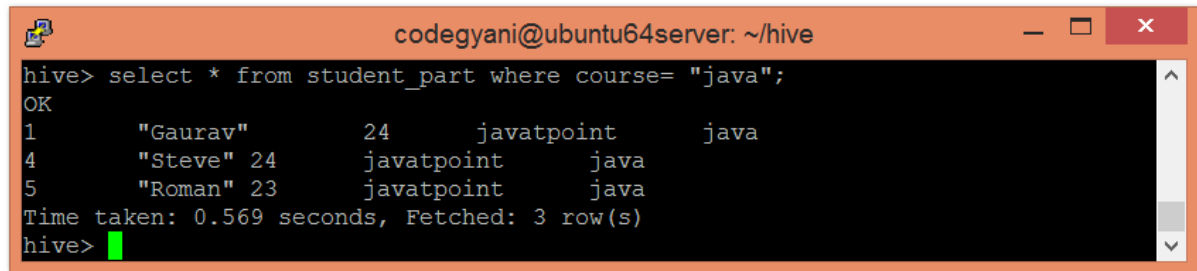
- Let's retrieve the entire data of the table by using the following command: -

hive> select * from student_part;

```
codegyani@ubuntu64server: ~/hive
hive> select * from student_part;
OK
NULL    NULL    NULL    NULL    __HIVE_DEFAULT_PARTITION__
2       "John"  22      javatpoint  hadoop
3       "William"  21      javatpoint  hadoop
1       "Gaurav"  24      javatpoint  java
4       "Steve"  24      javatpoint  java
5       "Roman"  23      javatpoint  java
Time taken: 1.231 seconds, Fetched: 6 row(s)
hive>
```

- Now, try to retrieve the data based on partitioned columns by using the following command: -

```
hive> select * from student_part where course= "java ";
```

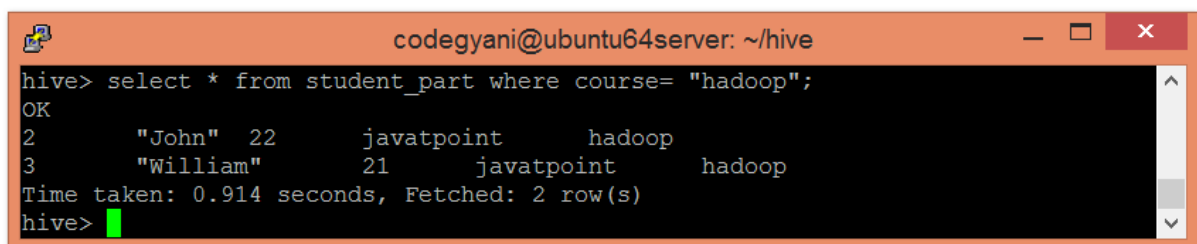
A terminal window titled 'codegyani@ubuntu64server: ~/hive' showing the execution of a Hive query. The query is 'select * from student_part where course= "java";'. The output shows three rows of data: ('Gaurav', 24, 'javatpoint', 'java'), ('Steve', 24, 'javatpoint', 'java'), and ('Roman', 23, 'javatpoint', 'java'). The time taken is 0.569 seconds and 3 rows were fetched. The prompt 'hive>' is followed by a green cursor.

```
codegyani@ubuntu64server: ~/hive
hive> select * from student_part where course= "java";
OK
1      "Gaurav"      24      javatpoint      java
4      "Steve" 24      javatpoint      java
5      "Roman" 23      javatpoint      java
Time taken: 0.569 seconds, Fetched: 3 row(s)
hive>
```

In this case, we are not examining the entire data. Hence, this approach improves query response time.

- Let's also retrieve the data of another partitioned dataset by using the following command: -

```
hive> select * from student_part where course= "hadoop";
```

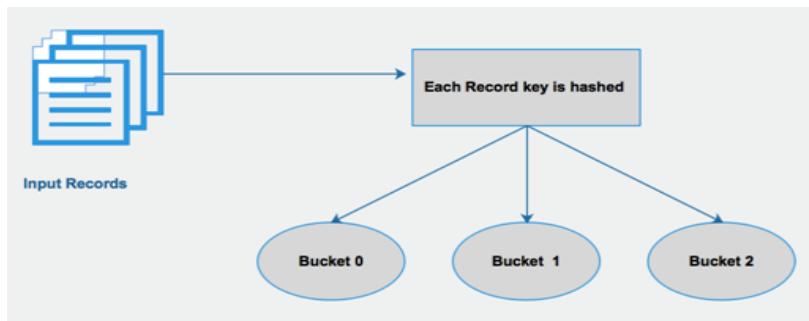
A terminal window titled 'codegyani@ubuntu64server: ~/hive' showing the execution of a Hive query. The query is 'select * from student_part where course= "hadoop";'. The output shows two rows of data: ('John', 22, 'javatpoint', 'hadoop') and ('William', 21, 'javatpoint', 'hadoop'). The time taken is 0.914 seconds and 2 rows were fetched. The prompt 'hive>' is followed by a green cursor.

```
codegyani@ubuntu64server: ~/hive
hive> select * from student_part where course= "hadoop";
OK
2      "John" 22      javatpoint      hadoop
3      "William" 21      javatpoint      hadoop
Time taken: 0.914 seconds, Fetched: 2 row(s)
hive>
```

Bucketing in Hive

The bucketing in Hive is a data organizing technique. It is similar to partitioning in Hive with an added functionality that it divides large datasets into more manageable parts known as buckets. So, we can use bucketing in Hive when the implementation of partitioning becomes difficult. However, we can also divide partitions further in buckets.

In bucketing, the partitions can be subdivided into buckets based on the hash function of a column.



- The concept of bucketing is based on the hashing technique.
- Here, modulus of current column value and the number of required buckets is calculated (let say, $F(x) \% 3$).
- Now, based on the resulted value, the data is stored into the corresponding bucket.

EXAMPLES:

- 1) Create a hive table
- 2) load the data in the hive table
- 3) create bucket table
- 4) set `hive.enforce.bucketing=true`;
- 5) load the data into bucket table.

PRACTICAL

```
Hive> select * from all_students;
```

```
Hive> create table students_buck(sno int,sname string ,passout_year string)  
clustered by(passout_year) sorted by (sno) into 3 buckets;
```

OK

Time taken:0.23 seconds

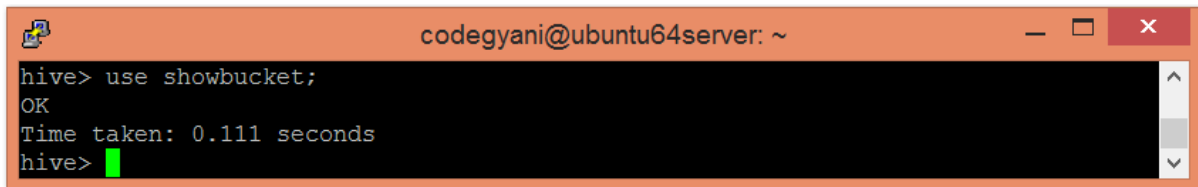
```
Hive> set hive.enforce.bucketing=true;
```

```
Hive> insert overwrite table students_buck select * from all_students;
```

Example of Bucketing in Hive

- First, select the database in which we want to create a table.

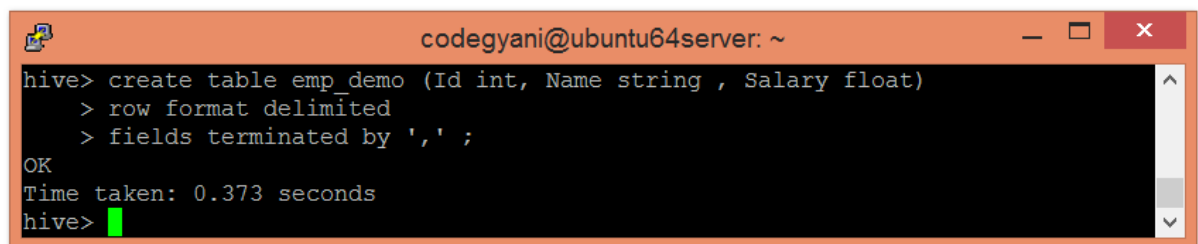
hive> use showbucket;

A terminal window titled 'codegyani@ubuntu64server: ~' showing the Hive command 'hive> use showbucket;'. The output is 'OK' and 'Time taken: 0.111 seconds'. The prompt 'hive>' is followed by a green cursor.

```
codegyani@ubuntu64server: ~
hive> use showbucket;
OK
Time taken: 0.111 seconds
hive>
```

- Create a dummy table to store the data.

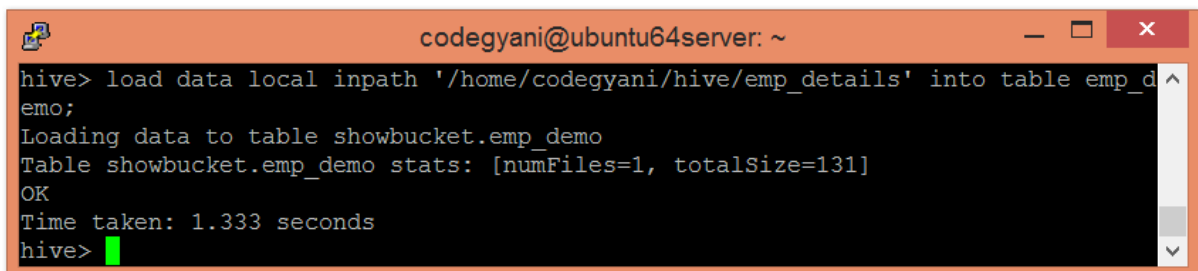
hive> create table emp_demo (Id int, Name string , Salary float)
row format delimited
fields terminated by ',';

A terminal window titled 'codegyani@ubuntu64server: ~' showing the Hive command 'hive> create table emp_demo (Id int, Name string , Salary float) row format delimited fields terminated by ',';'. The output is 'OK' and 'Time taken: 0.373 seconds'. The prompt 'hive>' is followed by a green cursor.

```
codegyani@ubuntu64server: ~
hive> create table emp_demo (Id int, Name string , Salary float)
> row format delimited
> fields terminated by ',' ;
OK
Time taken: 0.373 seconds
hive>
```

- Now, load the data into the table.

hive> load data local inpath '/home/codegyani/hive/emp_details' into table emp_demo;

A terminal window titled 'codegyani@ubuntu64server: ~' showing the Hive command 'hive> load data local inpath '/home/codegyani/hive/emp_details' into table emp_demo;'. The output is 'Loading data to table showbucket.emp_demo', 'Table showbucket.emp_demo stats: [numFiles=1, totalSize=131]', 'OK', and 'Time taken: 1.333 seconds'. The prompt 'hive>' is followed by a green cursor.

```
codegyani@ubuntu64server: ~
hive> load data local inpath '/home/codegyani/hive/emp_details' into table emp_demo;
Loading data to table showbucket.emp_demo
Table showbucket.emp_demo stats: [numFiles=1, totalSize=131]
OK
Time taken: 1.333 seconds
hive>
```

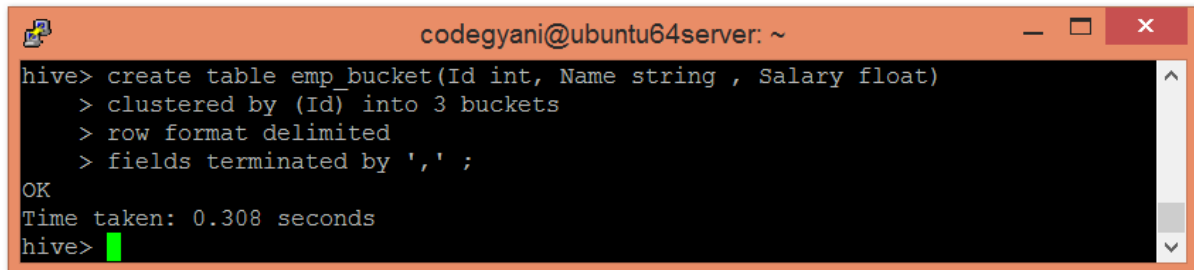
- Enable the bucketing by using the following command: -

hive> set **hive.enforce.bucketing** = **true**;

- Create a bucketing table by using the following command: -

hive> create table emp_bucket(Id int, Name string , Salary float)

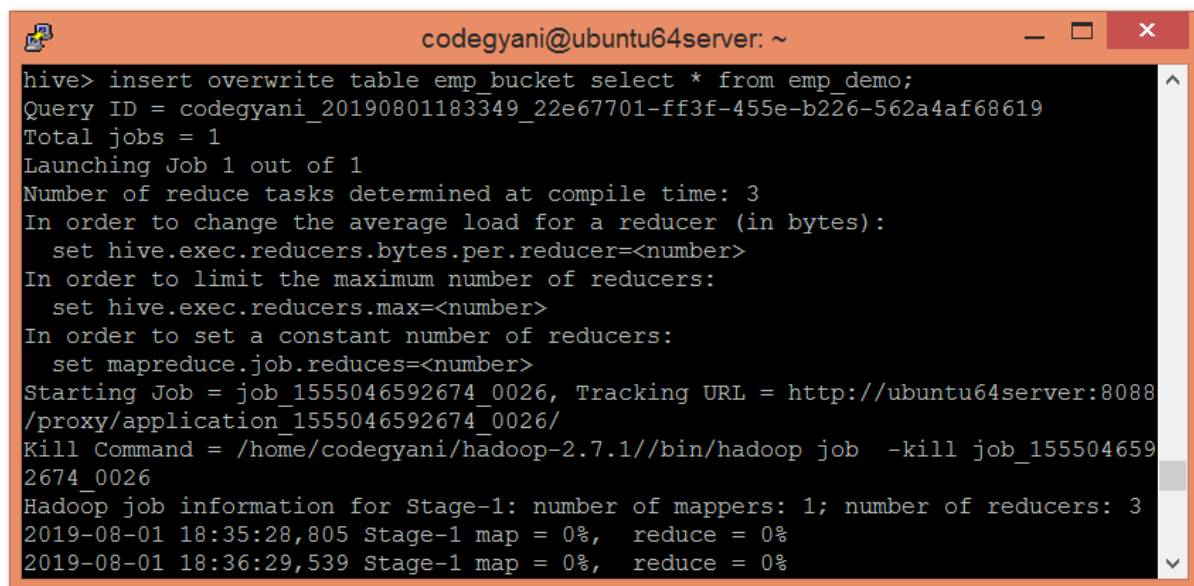
clustered by (Id) into 3 buckets
row format delimited
fields terminated by ',';

A terminal window titled 'codegyani@ubuntu64server: ~' with standard window controls. It shows the execution of a Hive command to create a table. The command is: 'hive> create table emp_bucket(Id int, Name string , Salary float) > clustered by (Id) into 3 buckets > row format delimited > fields terminated by ',' ;'. The output shows 'OK' and 'Time taken: 0.308 seconds'. The prompt 'hive>' is followed by a green cursor.

```
codegyani@ubuntu64server: ~
hive> create table emp_bucket(Id int, Name string , Salary float)
> clustered by (Id) into 3 buckets
> row format delimited
> fields terminated by ',' ;
OK
Time taken: 0.308 seconds
hive> █
```

- Now, insert the data of dummy table into the bucketed table.

hive> insert overwrite table emp_bucket select * from emp_demo;

A terminal window titled 'codegyani@ubuntu64server: ~' with standard window controls. It shows the execution of a Hive insert command and the subsequent Hadoop job progress. The command is: 'hive> insert overwrite table emp_bucket select * from emp_demo;'. The output includes job details like 'Query ID', 'Total jobs = 1', and 'Launching Job 1 out of 1'. It also shows configuration settings for reducers and the job's progress for Stage-1, indicating 0% completion for both map and reduce tasks.

```
codegyani@ubuntu64server: ~
hive> insert overwrite table emp_bucket select * from emp_demo;
Query ID = codegyani_20190801183349_22e67701-ff3f-455e-b226-562a4af68619
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1555046592674_0026, Tracking URL = http://ubuntu64server:8088/
proxy/application_1555046592674_0026/
Kill Command = /home/codegyani/hadoop-2.7.1/bin/hadoop job -kill job_1555046592674_0026
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 3
2019-08-01 18:35:28,805 Stage-1 map = 0%, reduce = 0%
2019-08-01 18:36:29,539 Stage-1 map = 0%, reduce = 0%
```

- Here, we can see that the data is divided into three buckets.

Hadoop Overview Datanodes Snapshot Startup Progress Utilities							
Browse Directory							
/user/hive/warehouse/showbucket.db/emp_bucket							Go!
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	codegyani	supergroup	56 B	8/2/2019, 4:11:20 AM	1	128 MB	000000_0
-rwxr-xr-x	codegyani	supergroup	53 B	8/2/2019, 4:11:20 AM	1	128 MB	000001_0
-rwxr-xr-x	codegyani	supergroup	54 B	8/2/2019, 4:11:20 AM	1	128 MB	000002_0

- Let's retrieve the data of bucket 0.

```
codegyani@ubuntu64server: ~
codegyani@ubuntu64server:~$ hdfs dfs -cat /user/hive/warehouse/showbucket.db/emp_bucket/000000_0;
\N,\N,\N
\N,\N,\N
6,"William",9000.0
3,"Vishal",40000.0
```

According to hash function :

$$7\%3=1$$

$$4\%3=1$$

$$1\%3=1$$

So, these columns stored in bucket 1.

- Let's retrieve the data of bucket 2.

```
codegyani@ubuntu64server: ~
codegyani@ubuntu64server:~$ hdfs dfs -cat /user/hive/warehouse/showbucket.db/emp_bucket/000002_0;
8,"Ronit",20000.0
5,"Henry",25000.0
2,"Aryan",20000.0
```

According to hash function :

$$8\%3=2$$

$$5\%3=2$$

$$2\%3=2$$

So, these columns stored in bucket 2.

In bucketing, splitting the data will be based upon rows.

In partitioning, splitting the data will be based upon on columns.

