Apache Hive

What is Hive

- Hive is a data warehouse infrastructure tool to process structured data in Hadoop.
- It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.
- Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.

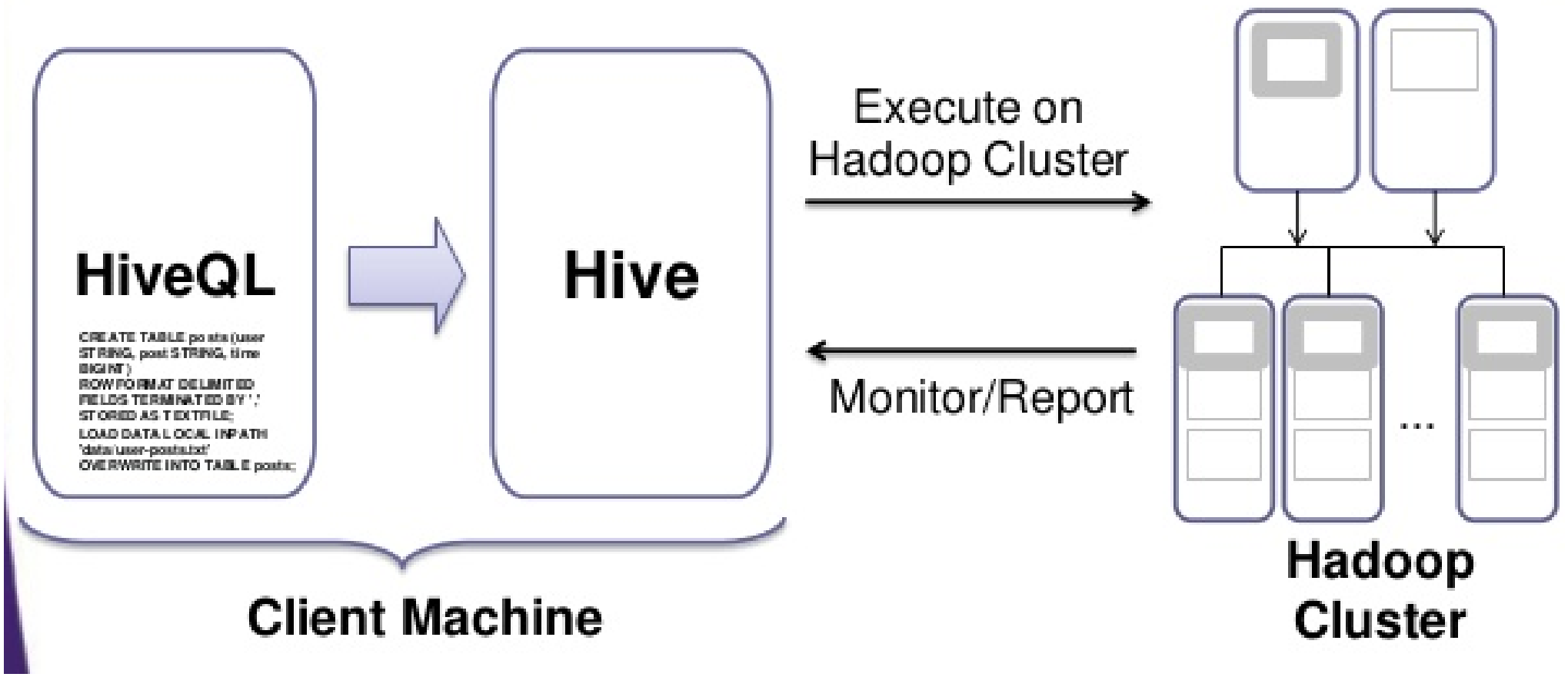
Features of Hive

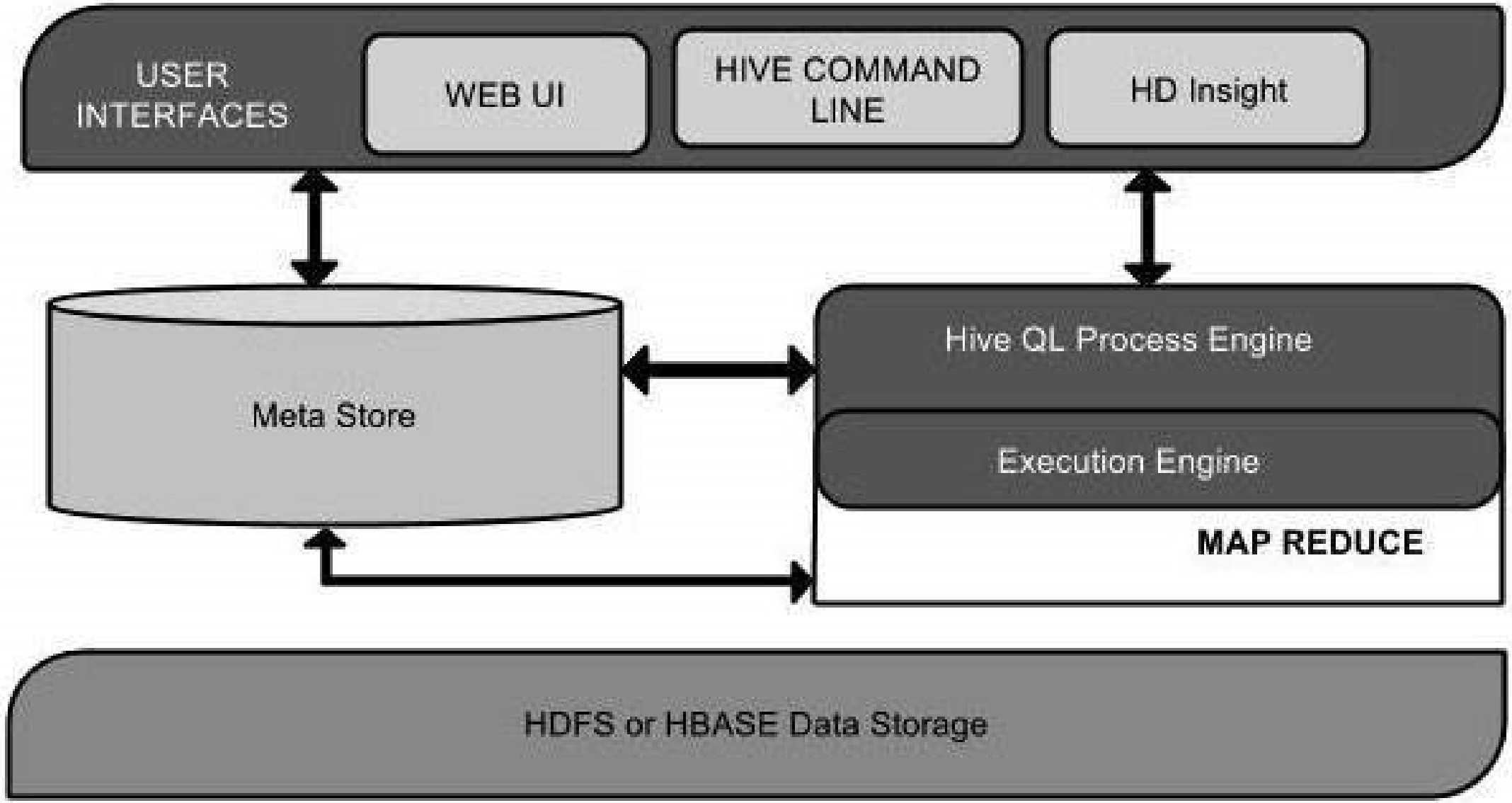
- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP not for OLTP.
- It provides SQL type language for querying called HiveQL or HQL.
- Hive is not RDBMS.
- It is familiar, fast, scalable, and extensible.

Characteristics of Hive

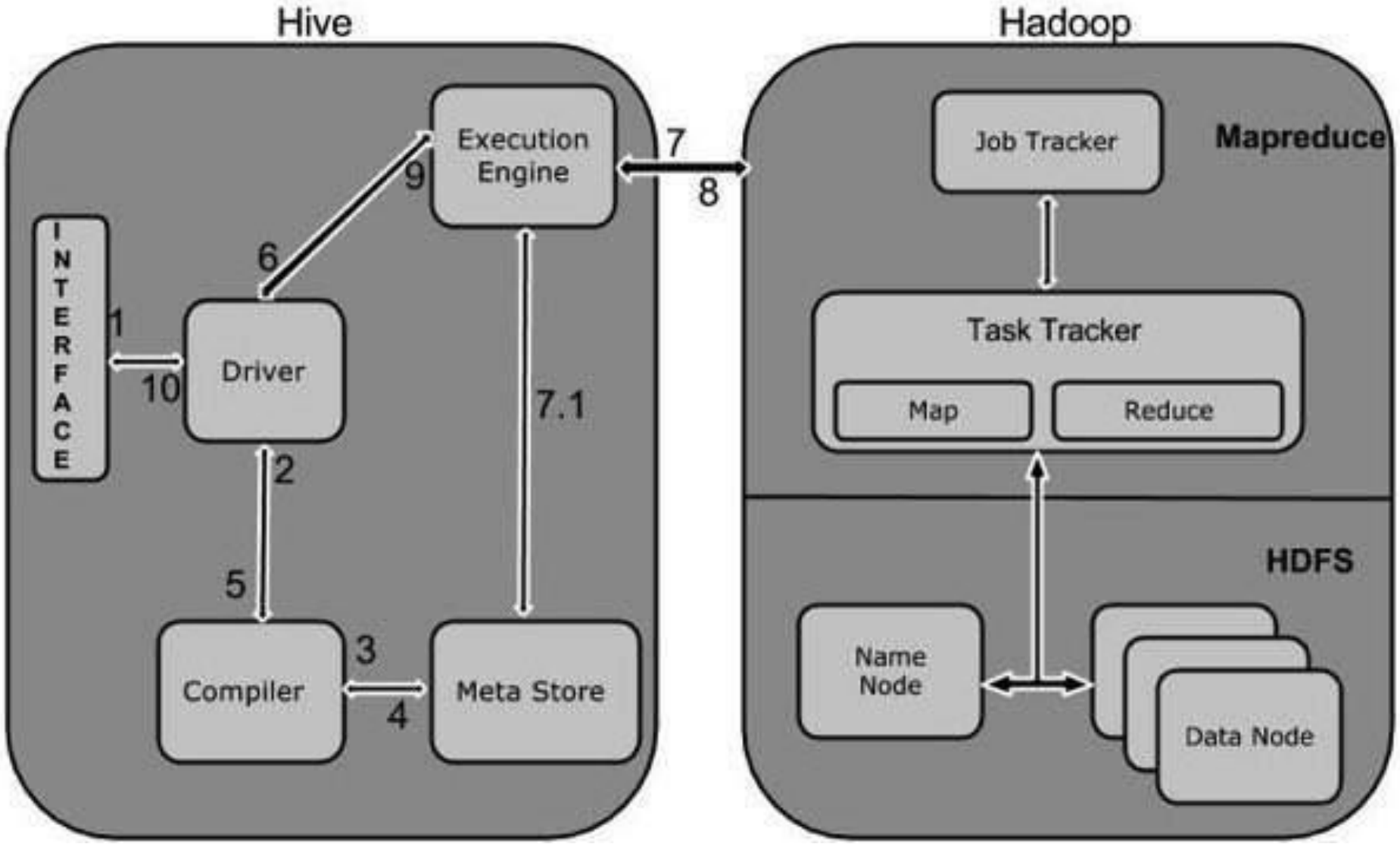
- Hive is a system for managing and querying un-structured data into structured format. It uses the concept of Map Reduce for execution.
- HDFS for storage and retrieval of data.

Hive Architecture




USER INTERFACES
WEB UI
HIVE COMMAND LINE
HD Insight
Meta Store
Hive QL Process Engine
Execution Engine
MAP REDUCE
HDFS or HBASE Data Storage

Hive interacts with Hadoop framework



Step No.	Operation
1	**Execute Query** The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
2	**Get Plan** The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
3	**Get Metadata** The compiler sends metadata request to Metastore (any database).
4	**Send Metadata** Metastore sends metadata as a response to the compiler.
5	**Send Plan** The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.

6	**Execute Plan** The driver sends the execute plan to the execution engine.
7	**Execute Job** Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
7.1	**Metadata Ops** Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8	**Fetch Result** The execution engine receives the results from Data nodes.
9	**Send Results** The execution engine sends those resultant values to the driver.
10	**Send Results** The driver sends the results to Hive Interfaces.

Hive Clients

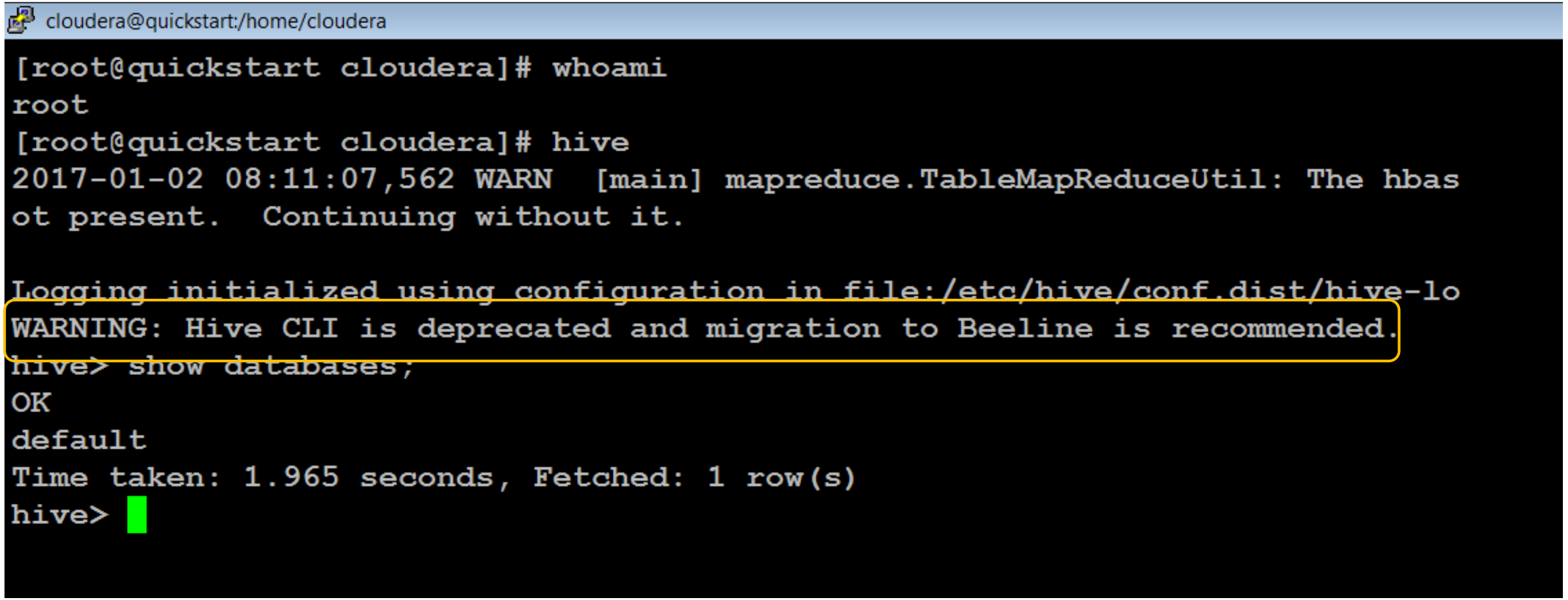
- Command Line Interfaces (CLI)
  - Hive CLI
  - Beeline
- Thick Clients (GUI)

  https://www.toadworld.com/m/freeware/1496

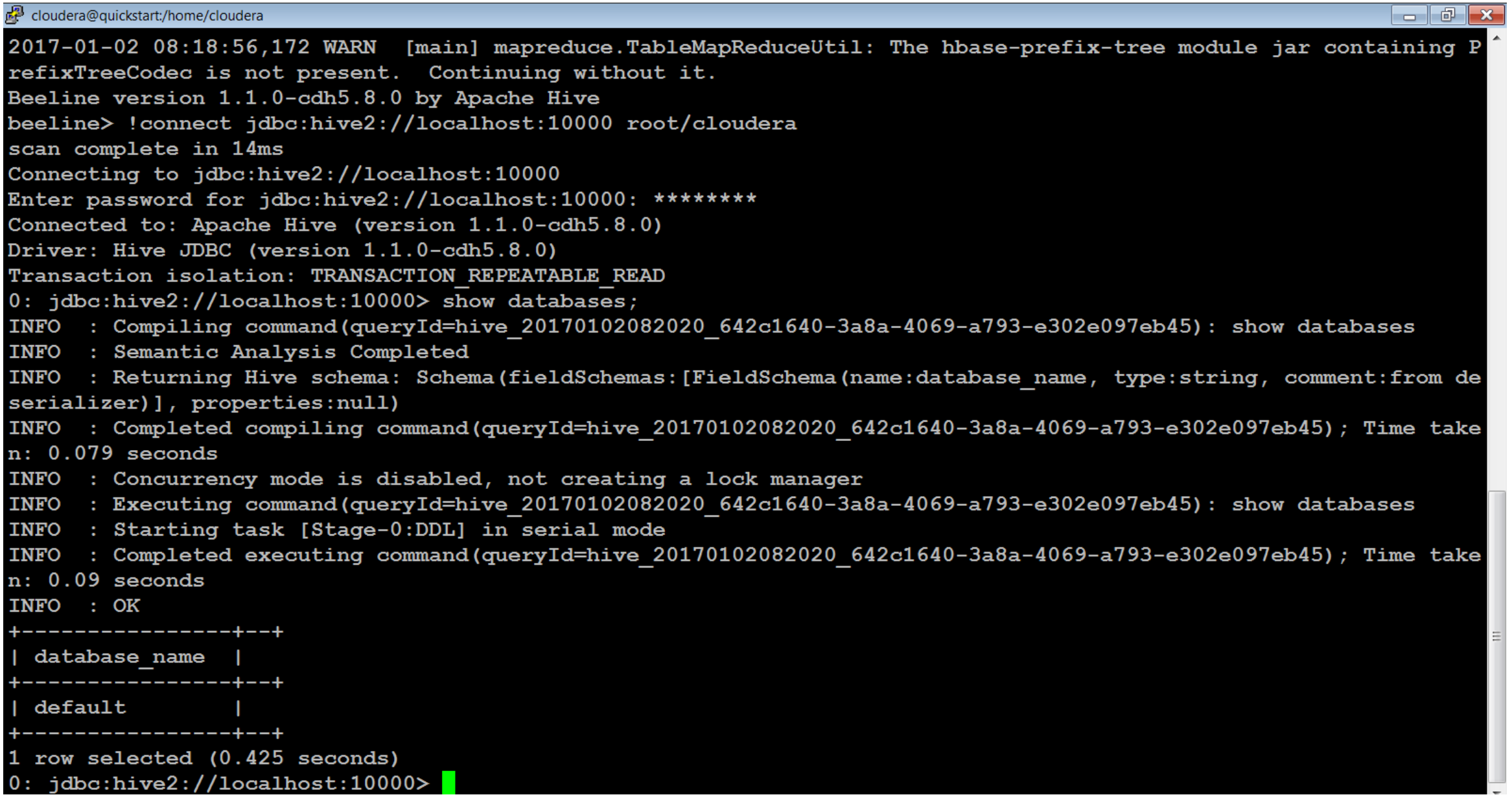
  https://sourceforge.net/projects/squirrel-sql/
- Web Browser (GUI)

  http://192.168.17.144:8888/

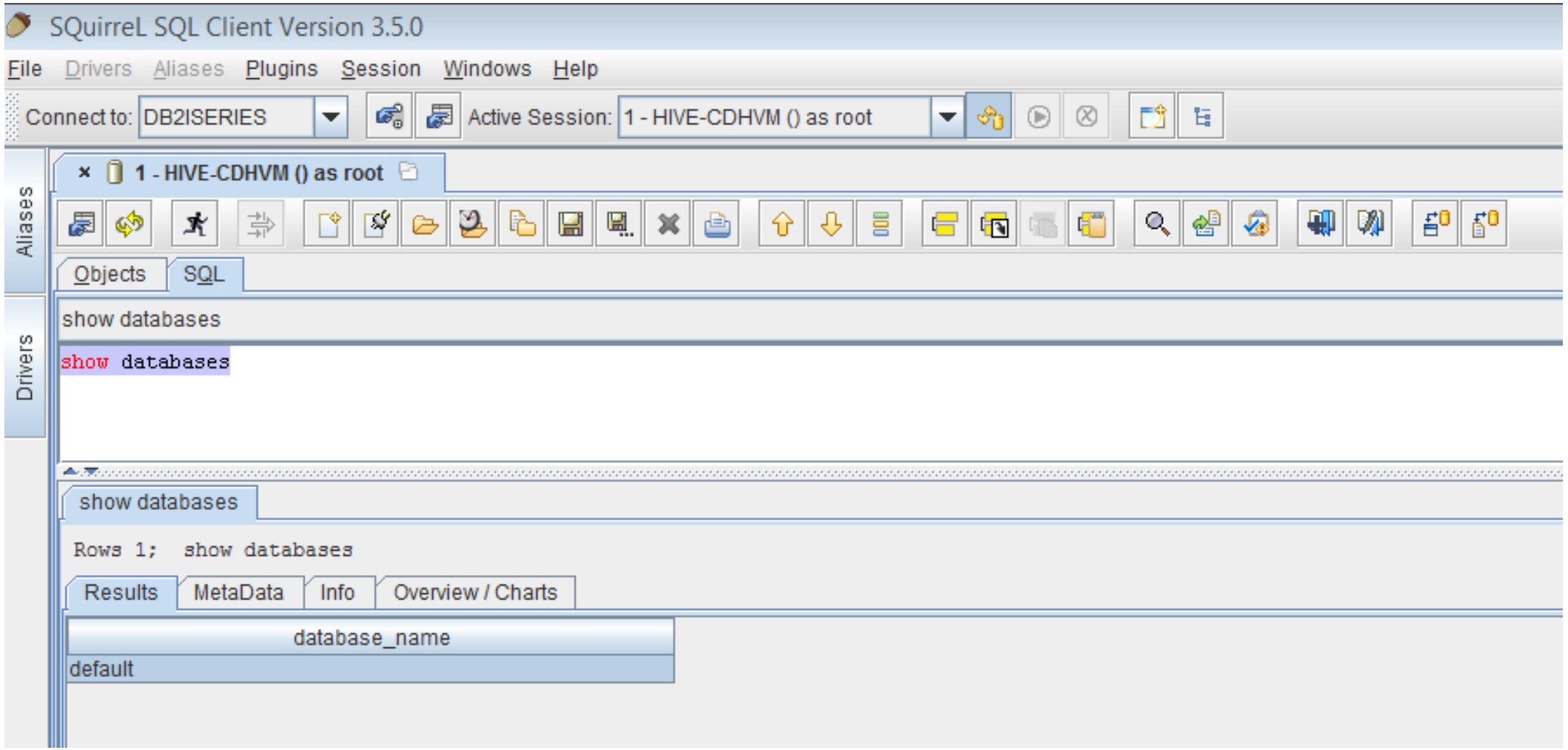
Hive CLI Client



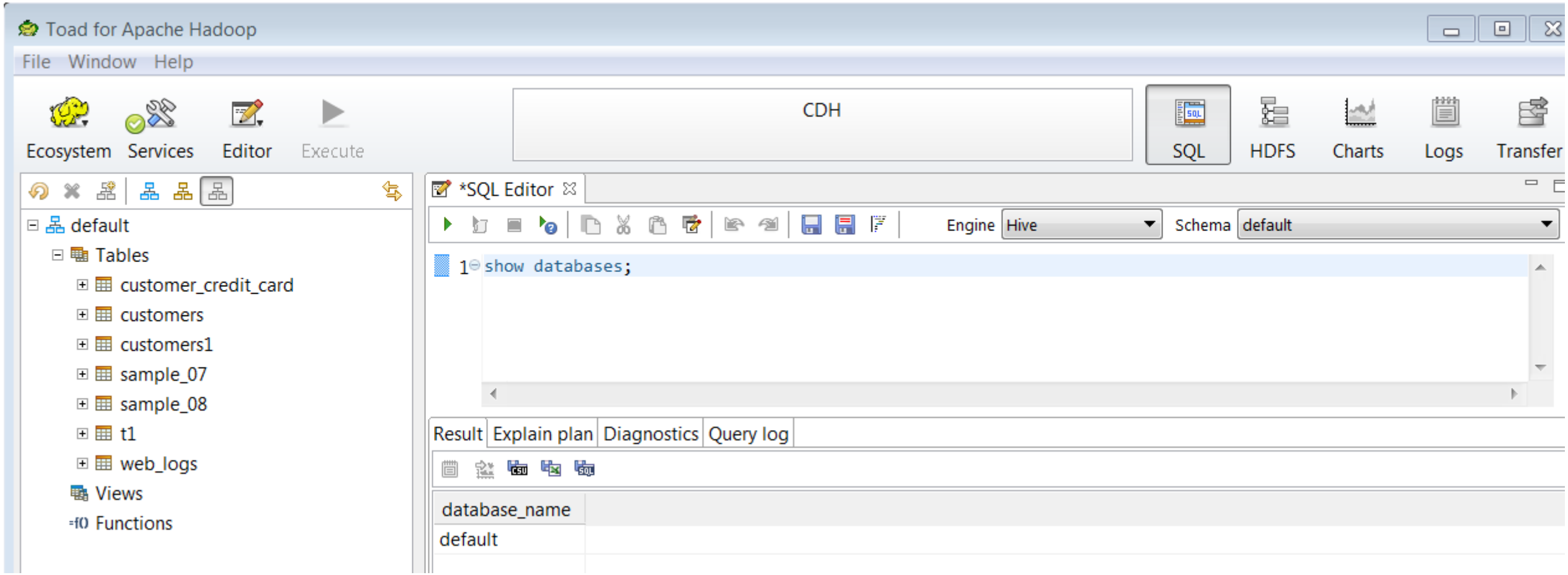
Beeline


cloudera@quickstart:/home/cloudera
2017-01-02 08:18:56,172 WARN [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing PrefixTreeCodec is not present. Continuing without it.
Beeline version 1.1.0-cdh5.8.0 by Apache Hive
beeline> !connect jdbc:hive2://localhost:10000 root/cloudera
scan complete in 14ms
Connecting to jdbc:hive2://localhost:10000
Enter password for jdbc:hive2://localhost:10000: ********
Connected to: Apache Hive (version 1.1.0-cdh5.8.0)
Driver: Hive JDBC (version 1.1.0-cdh5.8.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://localhost:10000> show databases;
INFO : Compiling command(queryId=hive_20170102082020_642c1640-3a8a-4069-a793-e302e097eb45): show databases
INFO : Semantic Analysis Completed
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:database_name, type:string, comment:from deserializer)], properties:null)
INFO : Completed compiling command(queryId=hive_20170102082020_642c1640-3a8a-4069-a793-e302e097eb45); Time taken: 0.079 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hive_20170102082020_642c1640-3a8a-4069-a793-e302e097eb45): show databases
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20170102082020_642c1640-3a8a-4069-a793-e302e097eb45); Time taken: 0.09 seconds
INFO : OK
+----------------+--+
| database_name |
+----------------+--+
| default |
+----------------+--+
1 row selected (0.425 seconds)
0: jdbc:hive2://localhost:10000>

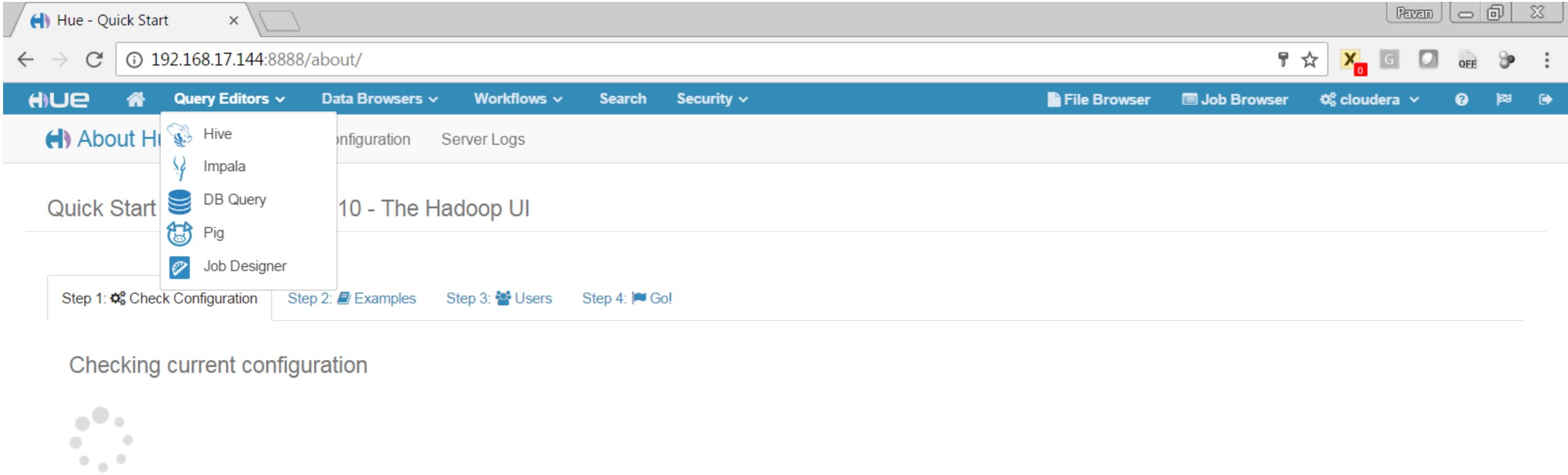
Squirrel SQL (JDBC Thick Client)



Toad for Hadoop Client



Hue Web UI


Hue - Quick Start
Pavan
192.168.17.144:8888/about/
Query Editors
Data Browsers
Workflows
Search
Security
File Browser
Job Browser
cloudera
Hive
Impala
DB Query
Pig
Job Designer
Server Logs
Quick Start
10 - The Hadoop UI
Step 1: Check Configuration
Step 2: Examples
Step 3: Users
Step 4: Go!
Checking current configuration

Starting and Stopping Hive

To start HiveServer2:

```
$ sudo service hive-server2 start
```

To stop HiveServer2:

```
$ sudo service hive-server2 stop
```

Creating Database/Schema

- Creating database:

  create database userdb; (or)

  create schema userdb;

- Listing existing databases:

  show databases;

- Pointing specific database:

  use database userdb;

Display current database:

  set hive.cli.print.current.db=true;

- Dropping database:

  drop database userdb; (or)

  drop schema userdb;

Hive Data types

- Numeric Types:
- Int, tiny int, big int, float, double, decimal
- String Types:
- Char, varchar, string
- Date/Time Types:
- Timestamp, date
- Complex Types
- Array, Map, Structs

Hive Tables

- **Internal tables**
- First we have to create table and load the data.
- We can call this one as **data on schema**.
- By dropping this table, both data and schema will be removed.
- The stored location of this table will be at /user/hive/warehouse.
- **When to Choose Internal Table:**
- Data is temporary
- Hive to Manage the table data completely not allowing any external source to use the table (external sources means sqoop, pig, mapreduce)
- Don’t want data after deletion

- **External tables**
- Data will be available in HDFS. The table is going to create on HDFS data.
- We can call this one as **schema on data.**
- At the time of dropping the table it drops only schema, the data will be still available in HDFS as before.
- External tables provide an option to create multiple schemas for the data stored in HDFS instead of deleting the data every time whenever schema updates
- **When to Choose External Table:**
- If processing data available in HDFS
- Useful when the files are being used outside of Hive

Working with Hive Internal (Managed) Tables

- Create a table and insert data.
- hive> create table employees (empid int,ename string,salary int,job string ,deptno int);

- hive> insert into employees values(101,'pavan',60000,'engineer',10);
- hive> insert into employees values(102,'kumar',50000,'manager',20);
- hive> select * from employees;

Browse Directory

/user/hive/warehouse/ Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	cloudera	supergroup	0 B	Mon Dec 19 21:15:06 -0800 2016	0	0 B	customers
drwxrwxrwx	root	supergroup	0 B	Fri Jan 06 23:42:14 -0800 2017	0	0 B	employees

- Create a table then load the data from local file system.

- Create table IF NOT EXISTS employee (eid int, name String,salary String, destination String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;

- LOAD DATA LOCAL INPATH '/home/cloudera/emp.txt' OVERWRITE INTO TABLE employee;

emp.txt

```
hive> Create table IF NOT EXISTS employee ( eid int, name String,salary String, destination String)ROW FORMAT DELIMITED FIELDS TERM
INATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;
OK
Time taken: 0.098 seconds
hive> LOAD DATA LOCAL INPATH '/home/cloudera/emp.txt' OVERWRITE INTO TABLE employee;
Loading data to table default.employee
Table default.employee stats: [numFiles=1, numRows=0, totalSize=164, rawDataSize=0]
OK
Time taken: 0.704 seconds
hive> select * from employee;
OK
1201    Gopal   45000   Technical manager
1202    Manisha 45000   Proof reader
1203    Masthanvali     40000   Technical writer
1204    Kiran   40000   Hr Admin
1205    Kranthi 30000   Op Admin
Time taken: 0.134 seconds, Fetched: 5 row(s)
hive>
```

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/user/hive/warehouse/employee Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxrwxrwx	root	supergroup	164 B	Sat Jan 07 00:03:59 -0800 2017	1	128 MB	emp.txt

- Create a table then load the data from HDFS.

- Copy emp.txt to HDFS

[root@quickstart cloudera]# hdfs dfs -put /home/cloudera/emp.txt /training/

- Create table IF NOT EXISTS employee (eid int, name String,salary String, destination String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;

- LOAD DATA INPATH '/training/emp.txt' OVERWRITE INTO TABLE employee;

root@quickstart:~

```
hive> Create table IF NOT EXISTS employee ( eid int, name String,salary String, destination String)ROW FORMAT DELIMITED FIELDS TERM
INATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;
OK
Time taken: 0.216 seconds
hive> LOAD DATA INPATH '/training/emp.txt' OVERWRITE INTO TABLE employee;
Loading data to table default.employee
Table default.employee stats: [numFiles=1, numRows=0, totalSize=164, rawDataSize=0]
OK
Time taken: 0.47 seconds
hive> select * From employee;
OK
1201    Gopal   45000   Technical manager
1202    Manisha 45000   Proof reader
1203    Masthanvali     40000   Technical writer
1204    Kiran   40000   Hr Admin
1205    Kranthi 30000   Op Admin
Time taken: 0.203 seconds, Fetched: 5 row(s)
```

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/user/hive/warehouse/employee Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxrwxrwx	root	supergroup	164 B	Sat Jan 07 00:17:11 -0800 2017	1	128 MB	emp.txt

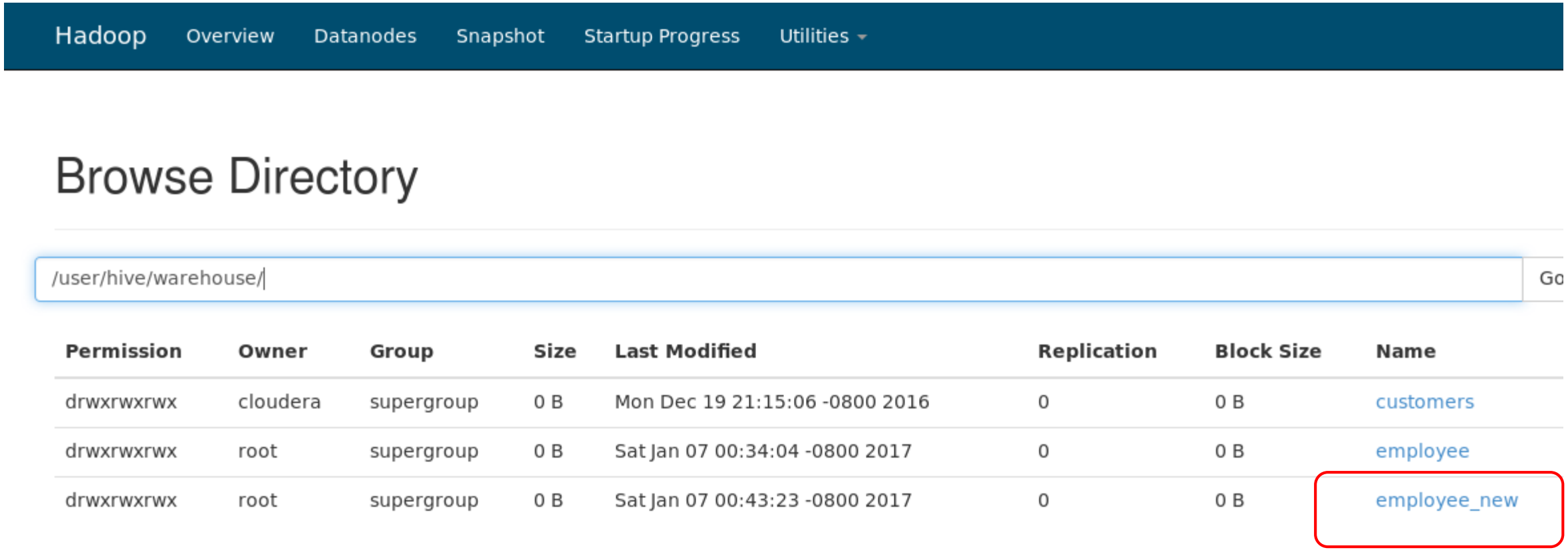
- Create a table based on another table
- hive> describe employee;

  eid int
  name string
  salary string
  destination string

- hive> select * From employee;

  1201 Gopal 45000 Technical manager
  1202 Manisha 45000 Proof reader
  1203 Masthanvali 40000 Technical writer
  1204 Kiran 40000 Hr Admin
  1205 Kranthi 30000 Op Admin

- hive> **create table** employee_new **as** select eid,name from employee;


Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Browse Directory
/user/hive/warehouse/
Go
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
drwxrwxrwx
cloudera
supergroup
0 B
Mon Dec 19 21:15:06 -0800 2016
0
0 B
customers
drwxrwxrwx
root
supergroup
0 B
Sat Jan 07 00:34:04 -0800 2017
0
0 B
employee
drwxrwxrwx
root
supergroup
0 B
Sat Jan 07 00:43:23 -0800 2017
0
0 B
employee_new

- Create a table then load the data from another table.
- create table empnames(names string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;

- insert into table empnames select name from employee_new;

Browse Directory

/user/hive/warehouse/

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	cloudera	supergroup	0 B	Mon Dec 19 21:15:06 -0800 2016	0	0 B	customers
drwxrwxrwx	root	supergroup	0 B	Sat Jan 07 00:43:23 -0800 2017	0	0 B	employee_new
drwxrwxrwx	root	supergroup	0 B	Sat Jan 07 00:51:59 -0800 2017	0	0 B	empnames

Working with Hive External Tables

- Create a External table and insert data.

- CREATE EXTERNAL TABLE EMPLOYEE_EXTERNAL(eid int, name String,salary String, destination String) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' LOCATION '/training/EMPLOYEE_EXTERNAL';

- insert into EMPLOYEE_EXTERNAL values(111,'pavan','50000','Technical manager');

- select * from EMPLOYEE_EXTERNAL;
- 111 pavan 50000 Technical manager

Browse Directory

/training Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	Sat Jan 07 01:44:33 -0800 2017	0	0 B	EMPLOYEE_EXTERNAL
-rw-r--r--	root	supergroup	105 B	Tue Jan 03 18:35:56 -0800 2017	1	128 MB	file1.txt

** If you drop the table , data file will not be removed from HDF.

- Create a table then load the data from local file system.

- hive> CREATE EXTERNAL TABLE EMPLOYEE_EXTERNAL(eid int, name String,salary String, destination String) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' LOCATION '/training/EMPLOYEE_EXTERNAL';

- hive> LOAD DATA LOCAL INPATH '/home/cloudera/emp.txt' OVERWRITE INTO TABLE employee_external;

- hive> LOAD DATA LOCAL INPATH '/home/cloudera/emp.txt' INTO TABLE employee_external;

Browse Directory

/training/EMPLOYEE_EXTERNAL Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	root	supergroup	164 B	Sat Jan 07 01:54:40 -0800 2017	1	128 MB	emp.txt
-rwxr-xr-x	root	supergroup	164 B	Sat Jan 07 01:55:05 -0800 2017	1	128 MB	emp_copy_1.txt

** If you use OVERWRITE in loading statement data will be override.
** Other wise copy of files will be created.
** If you drop the table , data file will not be removed from HDF.

- Create a External table then load the data from HDFS.

- hive> LOAD DATA INPATH '/training/emp.txt' INTO TABLE employee_external;

- hive> LOAD DATA INPATH '/training/emp.txt' INTO TABLE employee_external;

To Know schema of the table

- hive> describe employee;
- hive> describe extended employee;
- describe formatted employee;

Alter Table

- ALTER TABLE employee RENAME TO emp;
- ALTER TABLE employee CHANGE name ename String;
- ALTER TABLE employee CHANGE salary salary Double;
- ALTER TABLE employee ADD columns(name string comment “fistname”);

Hive Complex Data Types

array: It is an ordered collection of elements. The elements in the array must be of the same type.

map: It is an unordered collection of key-value pairs. Keys must be of primitive types. Values can be of any type.

struct: It is a collection of elements of different types.

Array

```
[cloudera@quickstart ~]$ cat arrrayfile.txt
    1,abc,40000,a$b$c,hyd
    2,def,3000,d$f,bang
```

```
hive> create table arrayTable(id int,name string,sal bigint,sub array<string>,city string)row format delimited fields terminated by ',' collection items terminated by '$';

hive> describe arrayTable;

hive> load data local inpath '/home/cloudera/arrrayfile.txt' into table arrayTable;

hive> select * from arrayTable;
    1    abc    40000   ["a","b","c"]   hyd
    2    def    3000    ["d","f"]       bang
hive>select sub[2] from arrayTable where id=1;

hive>select sub[0] from tarrayTable;
```

Map

[cloudera@quickstart ~]$ cat >mapfile.txt
1,abc,40000,abc,pf#500$epf#200,hyd
2,def,3000,d$f,pf#500,bang

hive> create table maptable(id int,name string,sal bigint,sub array<string>,dud map<string,int>,city string) row format delimited fields terminated by ',' collection items terminated by '$' map keys terminated by '#';

hive> load data local inpath '/home/cloudera/mapfile.txt' into table maptable;

hive> select * From maptable;

1	abc	40000	["a","b","c"]	{"pf":500,"epf":200}	hyd
2	def	3000	["d","f"]	{"pf":500}	bang

hive>select dud["pf"] from maptable;
hive> select dud["pf"],dud["epf"] from maptable;

Struct

[cloudera@quickstart ~]$ cat >structfile.txt

```
1,abc,40000,a$b$c,pf#500$epf#200,hyd$ap$500001
2,def,3000,d$f,pf#500,bang$kar$600038
```

```
hive> create table structtable(id int,name string,sal bigint,sub array<string>,dud map<string,int>,address struct<city:string,state:string,pin:bigint>) row format delimited fields terminated by ',' collection items terminated by '$' map keys terminated by '#';
```

```
hive> load data local inpath '/home/cloudera/structfile.txt' into table structtable;
hive> select * From structtable;
```

```
1   abc   40000   ["a","b","c"]   {"pf":500,"epf":200}   {"city":"hyd","state":"ap","pin":500001}
2   def   3000    ["d","f"]       {"pf":500}             {"city":"bang","state":"kar","pin":600038}
```

```
hive> select address.city from structtable;
```

XML Data Processing

- Create file with xml data:

cat sampxml1

```
<rec>
 <name>Ravi</name>
  <age>25</age>
  <gender>m</gender>
</rec>
<rec>
  <name>Rani</name>
  <gender>f</gender>
  <city>Hyd</city>
 </rec>
```

- Create a table with XML file:

  create table samp(line string);

- Loading data into table:

  load data local inpath '\cloudera\sampxml1' into table samp;

- select * from samp;
- select count(*) From samp;
- ***** above xml file has 2 records, when this file is loaded into hive table, hive treats, each line as a record,**
- ***as per xml two records.***
- ***as per hive 10 records(rows.)***

- cat sampxml2

  <rec><name>Ravi</name><age>25</age><city>hyd</city></rec>
  <rec><name>Rani</name><age>24</age><gender>f</gender></rec>
  <rec><name>Sampath</name><gender>m</gender><city>Del</city></rec>
- Create table samp2(line string);
- load data local inpath '/home/cloudera/sampxml2' into table samp2;
- select * from samp2;
- select count(*) from samp2
- select xpath_string(line,'rec/name') from samp2;
- select xpath_string(line,'rec/name'), xpath_int(line,'rec/age'), xpath_string(line,'rec/gender'), xpath_string(line,'rec/city') from samp2;

- **if string fields is missed, it returns blank string, if numeric field is missed it returns 0.

Hive Built-in Operators

- Relational Operators
- Arithmetic Operators
- Logical Operators

Relational Operators

Operator	Operand	Description
A = B	all primitive types	TRUE if expression A is equivalent to expression B otherwise FALSE.
A != B	all primitive types	TRUE if expression A is not equivalent to expression B otherwise FALSE.
A < B	all primitive types	TRUE if expression A is less than expression B otherwise FALSE.
A <= B	all primitive types	TRUE if expression A is less than or equal to expression B otherwise FALSE.
A > B	all primitive types	TRUE if expression A is greater than expression B otherwise FALSE.
A >= B	all primitive types	TRUE if expression A is greater than or equal to expression B otherwise FALSE.
A IS NULL	all types	TRUE if expression A evaluates to NULL otherwise FALSE.
A IS NOT NULL	all types	FALSE if expression A evaluates to NULL otherwise TRUE.

Arithmetic Operators

Operators	Operand	Description
A + B	all number types	Gives the result of adding A and B.
A - B	all number types	Gives the result of subtracting B from A.
A * B	all number types	Gives the result of multiplying A and B.
A / B	all number types	Gives the result of dividing B from A.
A % B	all number types	Gives the reminder resulting from dividing A by B.
A & B	all number types	Gives the result of bitwise AND of A and B.
A | B	all number types	Gives the result of bitwise OR of A and B.
~A	all number types	Gives the result of bitwise NOT of A.

Local Operators

Operators	Operands	Description
A AND B	Boolean	TRUE if both A and B are TRUE, otherwise FALSE.
A && B	Boolean	Same as A AND B.
A OR B	Boolean	TRUE if either A or B or both are TRUE, otherwise FALSE.
A || B	Boolean	Same as A OR B.
NOT A	Boolean	TRUE if A is FALSE, otherwise FALSE.
!A	boolean	Same as NOT A.

Hive Built-in functions

Adobe Acrobat Document

Hive Views

- Creating view

```
hive> CREATE VIEW emp_30000 AS
SELECT * FROM employee
WHERE salary>30000;
```

- Dropping a view

```
hive> DROP VIEW emp_30000;
```

Hive Indexes

- Creating a Index

hive> CREATE INDEX inedx_salary ON TABLE employee(salary) AS 'compact' WITH DEFERRED REBUILD;

- Show index
- show index on employee;
- Dropping an index

hive> DROP INDEX index_salary ON employee;

HQL - Select-Where

- The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore.

- Generate a query to retrieve the employee details who earn a salary of more than Rs 30000.

```
hive> SELECT * FROM employee WHERE salary>30000;
```

HQL - Select-Order By

- The ORDER BY clause is used to retrieve the details based on one column and sort the result set by ascending or descending order.

- Generate a query to retrieve the employee details in order by using Department name.

hive> SELECT eid, name, destination FROM employee ORDER BY eid;

HQL - Select-Group By

- The GROUP BY clause is used to group all the records in a result set using a particular collection column. It is used to query a group of records.

- Generate a query to retrieve the number of employees in each department.

- hive> SELECT Dept,count(*) FROM employee GROUP BY DEPT;

Sample Tables


owners.txt


cars.txt

- Create tables
- create table owners(ID int,Name string,Age int,Car_ID int)row format delimited fields terminated by ',' lines terminated by'\n';
- create table cars(ID int,Make String,Model String,Year string)row format delimited fields terminated by',' lines terminated by '\n';
- Load the data into tables
- LOAD DATA LOCAL INPATH '/home/cloudera/owners.txt' OVERWRITE INTO TABLE owners;
- LOAD DATA LOCAL INPATH '/home/cloudera/cars.txt' OVERWRITE INTO TABLE cars;

Aggregation Functions available in Hive

- count(*) & count(expr)
- count(*) - Returns the total number of rows, including NULL values.
- count(expr) - Returns the number of rows for which the supplied expression is non-NULL.
- sum(col) & sum(DISTINCT col)
- sum(col) - Returns the sum of the elements in the group or the
- sum(DISTINCT col) - sum of the distinct values of the column in the group.

- avg(col) & avg(DISTINCT col)
- Returns the average of the elements in the group or the average of the distinct values of the column in the group.
- min(col) - Returns the minimum of the column in the group.
- max(col) - Returns the maximum value of the column in the
- group.
- variance(col), var_pop(col) - Returns the variance of a numeric column in the group.

Filtering Aggregate Results with the HAVING Clause

- We only want to see the car makers that have more than 1 model in our table.

- SELECT make, count(distinct model) FROM cars GROUP BY make HAVING count(distinct model) > 1;

LIMIT Results

- Restrict the number of rows returned from a result set.
- SELECT make, model FROM cars LIMIT 2;

- Union, Intersect & Minus doesn’t support in Hive.

- Intersect we can achieve using join.
- select distinct cars.* from cars inner join vehicles on cars.id = vehicles.id;
- Minus also we can achieve using join.
- select distinct cars.* from cars left outer join vehicles on cars.id = vehicles.id where vehicles.year is null;
- select distinct cars.* from cars right outer join vehicles on cars.id = vehicles.id where vehicles.year is null;

UNION ALL

- The UNION ALL command allows you to concatenate two tables with the same schema together into one table.

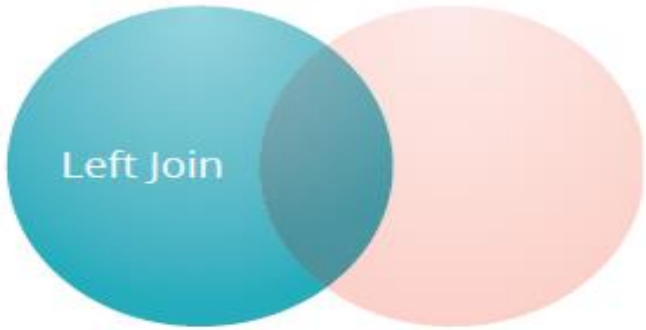
- select * from cars union all select * from vehicles;

HiveQL - Joins

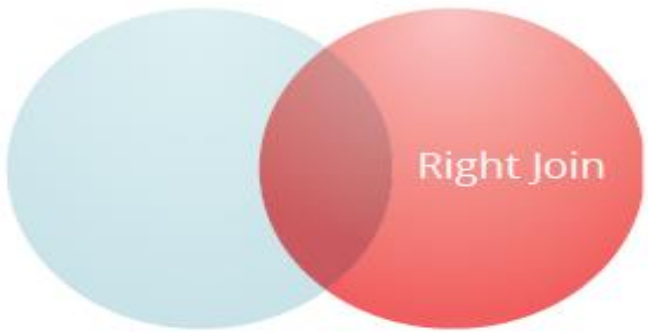
- JOIN is a clause that is used for combining specific fields from two tables by using values common to each one. It is used to combine records from two or more tables in the database. It is more or less similar to SQL JOIN.
- Types:
- JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

Adobe Acrobat Document


```
SELECT a.coll, ... a.coln, b.coll, ... b.coln
FROM a
JOIN b ON (a.id = b.id);
```




```
SELECT a.coll, ... a.coln, b.coll, ... b.coln
FROM a
LEFT OUTER JOIN b ON (a.id = b.id);
```




```
SELECT a.coll, ... a.coln, b.coll, ... b.coln
FROM a
RIGHT OUTER JOIN b ON (a.id = b.id);
```


```
SELECT a.coll, ... a.coln, b.coll, ... b.coln
FROM a
FULL OUTER JOIN b ON (a.id = b.id);
```

Sample Schemas for Joins


orders.txt


customers.txt

- create table customers(ID int,Name string,Age int,Address String,Salary int) row format delimited fields terminated by ',';
- create table orders(oid int,date date,customer_id int,amount int) row format delimited fields terminated by ',';

- LOAD DATA LOCAL INPATH '/home/cloudera/customers.txt' OVERWRITE INTO TABLE customers;
- LOAD DATA LOCAL INPATH '/home/cloudera/orders.txt' OVERWRITE INTO TABLE orders;

- INNER JOIN:
- SELECT c.ID, c.NAME, c.AGE, o.AMOUNT FROM CUSTOMERS c JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);
- LEFT OUTER JOIN
- SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c LEFT OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);
- RIGHT OUTER JOIN
- SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c RIGHT OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);
- FULL OUTER JOIN
- SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c FULL OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);

Hive –Bucketing

- **Bucketing** decomposes data into more manageable or equal parts. With partitioning, there is a possibility that you can create multiple small partitions based on column values. If you go for **bucketing**, you are restricting number of buckets to store the data. This number is defined during table creation scripts

Bucketing example

- **Creating a table:**
- create table studentdata(name string,email string,contactno bigint,exp string) row format delimited fields terminated by ',';
- **Loading data from csv file:**
- load data local inpath '/home/cloudera/complexdata.csv' into table studentdata;
- **creating bucket table:**
- create table buck(name string,email string,contactno bigint,exp string) clustered by (exp) sorted by (contactno) into 5 buckets;
- **Enable bucketing table:**
- set hive.enforce.bucketing=true;
- **Insert data into bucket table:**
- insert overwrite table buck select * from studentdata;
- *** After bucketing splits will be stored in /user/hive/warehouse/buck*
- **Test Bucket table:**
- select * from buck tablesample(bucket 3 out of 5 on exp);
- **Test normal table:**
- select * from studentdata limit 10;

Microsoft Excel

ma Separated Valu