

# HIVE

---

Senthil Kumar A



# Introduction

- An Abstraction on top of MapReduce
- Allows users to query data in the Hadoop cluster without knowing Java or MapReduce
- Structured Data in HDFS logically into Tables
- Uses the HiveQL Language
  - Very similar to SQL
  - Turns HiveQL into MapReduce Jobs

# Installation

- Download <http://hive.apache.org>
- `tar -xvf hive-x.y.z-dev.tar.gz`
- `export HADOOP_HOME=/<HADOOP DIR>`
- `bin/hive`

www.datadotz.com

# Create database

- bin/hive
- show databases;
- create database training;
- show databases;
- use training;
- show tables;

www.datadotz.com

# Hive Shell - Examples

- bin/hive
- show tables;
- create table patient(pid INT, pname STRING, drug STRING,gender STRING, tot\_amt INT) row format delimited fields terminated by ',' stored as textfile;
- load data inpath '/home/senthil/pfile.txt' into table patient;
- select count(\*) from patient;
- select sum(tot\_amt) from patient where drug = 'paracetamol';
- select max(tot\_amt) from patient group by drug;
- show tables;
- desc patient;
- desc extended patient;
- show functions;
- insert overwrite local directory '/home/senthil/results' select \* from patient;
- select \* from patient where drug in ('avil','metacin');

```
01,senthil,paracetamol,male,12
02,saravanan,avil,male,44
03,rajesh,metacin,male,26
04,usha,paracetamol,female,20
05,alex,paracetamol,male,48
06,nasir,metacin,male,37
07,singh,paracetamol,male,15
```

## Some more..

```
paracetamol  
avil  
metacin
```

- create table drug(drugname STRING) row format delimited fields terminated by ',' stored as textfile;
- load data local inpath '/home/senthil/drug\_file.txt' into table drug;
- select \* from patient join drug on patient.drug=drug.drug;
- select patient.\* from patient left outer join drug on patient.drug = drug.drug where drug.drug is NULL;
- create table drug\_new(drug STRING) row format delimited fields terminated by ',' stored as textfile;
- insert overwrite table drug\_new select \* from drug;
- select \* from drug\_new;
- insert into table drug\_new select \* from drug;
- select \* from drug\_new;

# External Table

- create EXTERNAL table patient\_external(pid INT, pname STRING, drug STRING,gender STRING,tot\_amt INT) row format delimited fields terminated by ',' stored as textfile LOCATION '/patient\_external';
- LOAD DATA INPATH '/data10.txt' INTO table patient\_external;
- drop table patient\_external;
- "chech the table in hdfs path"

# Partitions

- A way of dividing table into multiple parts based upon a column value such as Date
- defined at table creation time using the PARTITIONED BY clause
- *create table patient (pid INT, pname STRING, drug STRING, tot\_amt INT) partitioned by (dt STRING, country STRING) row format delimited fields terminated by ';' stored as textfile;*
- *LOAD DATA LOCAL INPATH '/tmp/file\_ind.txt' INTO TABLE logs PARTITION (dt='2012-11-01', country='IND');*
- Above code creates a sub partition called country in date
- Please look at the directory structure in the HDFS
- Our data in the files should not contain the columns used for partitioning



# Partition queries

- create table patient\_partition\_1(pid INT, pname STRING, drug STRING,gender STRING,tot\_amt INT)partitioned by(country STRING)row format delimited fields terminated by ',' stored as textfile;
- LOAD DATA LOCAL INPATH '/home/username/Desktop/patient\_file.txt' INTO TABLE patient\_partition\_1 PARTITION (country='IND');
- **Check the /user/hive/warehouse/patient\_partition\_1 directory in HDFS**
- create table patient\_partition\_2 (pid INT, pname STRING, drug STRING,gender STRING,tot\_amt INT) partitioned by (dt STRING, country STRING)row formatdelimited fields terminated by ',' stored as textfile;
- desc patient\_partition\_2;
- LOAD DATA LOCAL INPATH '/home/username/Desktop/patient\_file.txt' INTO TABLE patient\_partition\_2 PARTITION (dt='2012-11-01', country='IND');
- **Check the /user/hive/warehouse/ patient\_partition\_2**
- Select \* from patient\_partition\_1;
- Select \* from patient\_partition\_2;

# Buckets

- To enable map side join effectively
- To do sampling
- *CREATE TABLE bucketed\_patient (pid INT, pname STRING, drug STRING, genter STRING, tot\_amt INT) CLUSTERED BY (pid) INTO 4 BUCKETS;*
- *hive.enforce.bucketing = true;*
- *INSERT OVERWRITE TABLE bucketed\_patient SELECT \* FROM patient;*
- *Can contain multiple fields for bucketing.*

# Partition with Bucket

- `CREATE TABLE partition_bucketed(pid INT, pname STRING, drug STRING, gender STRING, tot_amt STRING)PARTITIONED BY(country STRING)CLUSTERED BY(pid) INTO 4 BUCKETS;`
- `set hive.enforce.bucketing = true;`
- `LOAD DATA LOCAL INPATH '/home/saravanan/Desktop/datagen_10.txt' INTO TABLE partition_bucketed PARTITION (country='IND');`
- `INSERT OVERWRITE TABLE partition_bucketed PARTITION (country='us') SELECT * FROM patient;`
- `select * from partition_bucketed where country= 'IND';`
- `select * from partition_bucketed where country= 'us';`

# Some more queries

- Alter Table patient RENAME TO patient\_new;
- ALTER TABLE patient\_new ADD COLUMNS (extra STRING);
- Sub Queries
- Views

www.datadotz.com

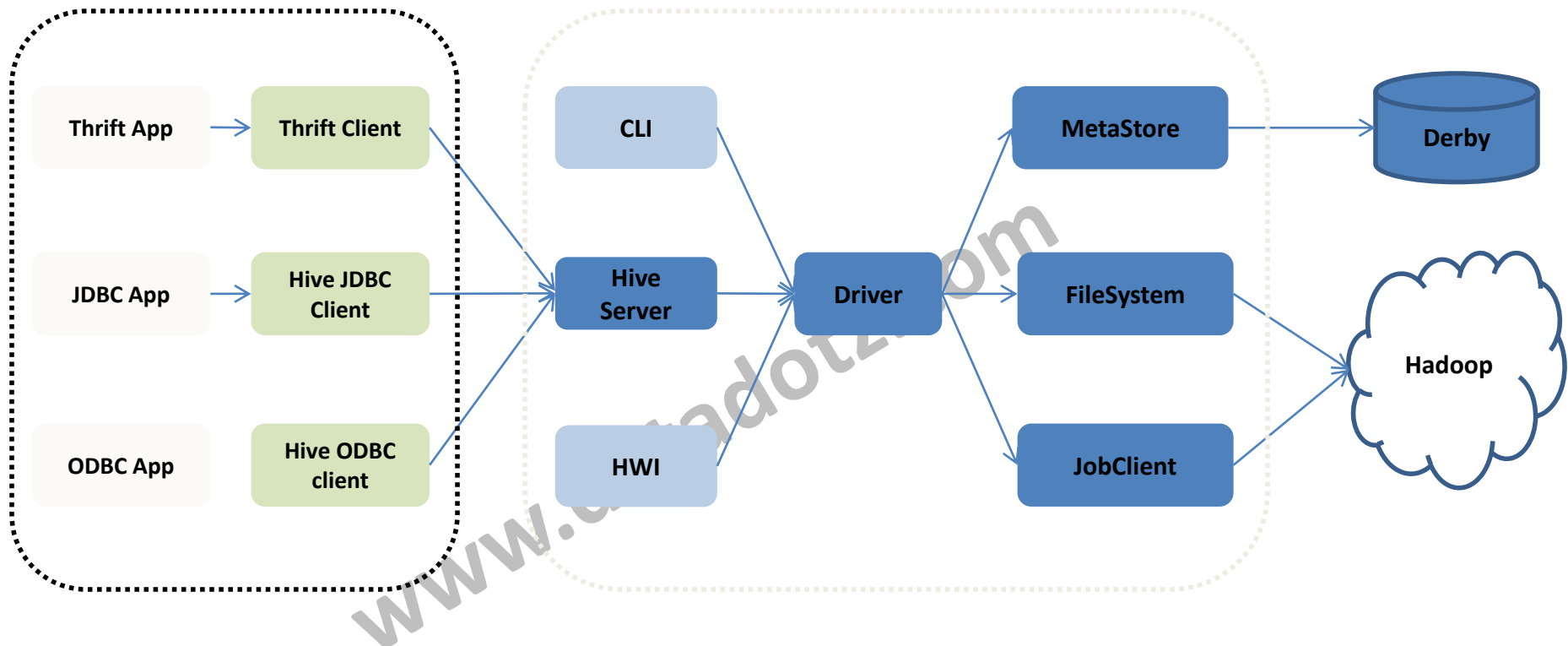
# Data Types

- Primitive types
  - Integers: TINYINT, SMALLINT, INT, BIGINT.
  - Boolean: BOOLEAN.
  - Floating point numbers: FLOAT, DOUBLE .
  - String: STRING.
- Complex types
  - Structs: {a INT; b INT}.
  - Maps: M['group'].
  - Arrays: ['a', 'b', 'c'], A[1] returns 'b'.

**Date Format ?**



# Hive Architecture



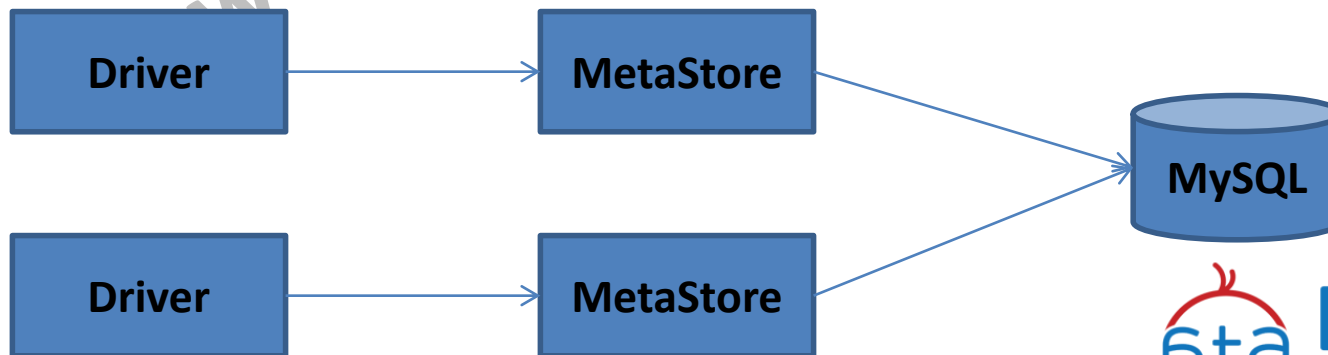
# MetaStore

*central repository of Hive metadata*

## Embedded MetaStore



## Local MetaStore



# JOINS

- `SELECT a.* FROM a JOIN b ON (a.id = b.id)`
- Join on multiple columns
  - `SELECT a.* FROM a JOIN b ON (a.id = b.id AND a.department = b.department)`
- Join on Multiple tables
  - `SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key1)`
  - `SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key2)`
- LEFT, RIGHT, and FULL OUTER joins – available in HiveQL
  - `SELECT a.val, b.val FROM a LEFT OUTER JOIN b ON (a.key=b.key)`
  - provides more control over ON clauses for which there is no match
- WHERE clause available using JOIN
- `SELECT a.val, b.val FROM a LEFT OUTER JOIN b ON (a.key=b.key) WHERE a.ds='2009-07-07' AND b.ds='2009-07-07'`



# Built in operators

- Arithmetic Operators
  - +, -, \*, /, %, |, ^, ~
- Relational Operators
  - =, <=>, !=, <>, <, >, <=, >=, IS NULL, IS NOT NULL, LIKE, RLIKE, REGEXP, BETWEEN, NOT BETWEEN
- Logical Operators
  - AND, OR, &&, ||, NOT, !

# Built-in Functions

- Mathematical: round, floor, ceil, rand, exp...
- Collection: size, map\_keys, map\_values, array\_contains.
- Type Conversion: cast.
- Date: from\_unixtime, to\_date, year, datediff...
- Conditional: if, case, coalesce.
- String: length, reverse, upper, trim...

# Hive Web Interface

- Configuration - hive-site.xml
  - hive.hwi.listen.port -- 0.0.0.0
  - hive.hwi.listen.host -- 9999
  - hive.hwi.war.file -- /lib/hive\_hwi.war
- bin/hive --service hwi
- Features
  - Schema Browsing
  - Detached query execution
  - No local installation
  - Results are stored locally in hive server machine

# Hive JDBC

- bin/hive –service hiveserver
- uri is just “jdbc:hive://localhost:10000/default”



# Other clients

- ODBC
- Thrift Clients
  - A software framework, for scalable cross-language services development, combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml and Delphi and other languages
  - Python, PHP, Java, C++

# Custom User Defined Functions

```
package com.example.hive.udf;  
import org.apache.hadoop.hive.ql.exec.UDF;  
import org.apache.hadoop.io.Text;  
public final class Lower extends UDF  
{  
    public Text evaluate(final Text s){  
        if (s == null)  
            {return null; }  
        return new Text(s.toString().toLowerCase());  
    }  
}
```

**UDF operates on a single row and produces a single row as its output.**

## UDF continued...

- hive> add jar my\_jar.jar;
- hive> list jars;
- hive> create temporary function my\_lower as 'com.example.hive.udf.Lower';
- Hive> select my\_lower(title), sum(freq) from titles group by my\_lower(title);

# UDAF

- works on multiple input rows
- It may either creates a single output row or create a multiple output rows
- Methods to override
  - init, iterate, terminatePartial, merge, terminate



# Storage Formats

- Default
  - delimited - Control-A character with a row per line
- Two dimensions – row format & file formats
- Use ROW FORMAT or STORED AS for the above two
- ROW format uses Serde
  - Serde – serializers and deserializers
- Binary FileFormats
  - SequenceFile
  - RCFILE
    - Default in industry
    - ***CREATE TABLE ... ROW FORMAT SERDE***  
***'org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe' STORED AS RCFILE;***

# RC file format

- create table patient\_new(pid INT, pname STRING, drug STRING,gender STRING,tot\_amt INT) row format delimited fields terminated by ',' stored as textfile;
- load data local inpath '/home/saravanan/Desktop/data\_30l.txt' into table patient\_new;
- create table patient\_rc(pid INT, pname STRING, drug STRING,gender STRING,tot\_amt INT) ROW FORMAT SERDE'org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe' STORED AS RCFILE;
- insert overwrite table patient\_rc select \* from patient\_new;
- select pid from patient\_new where tot\_amt =110;
- select pid from patient\_rc where tot\_amt =110;
- Look at the time difference between two.

# Map Reduce Scripts in Hive

- add file bin/test.py;
- Insert overwrite local directory '/tmp/result.txt' **transform** (rx.id) using 'python test.py' as (k,v) from (select pid as id from patient ) rx;
- Similarly you can use **MAP** and **REDUCE**
- *Please refer to the example code available*

# Hive – Hbase integration

- Create a Hbase table “testtable” with column family “data” and column Qualifies “name”
- Replace the hbase-\*.jar , zookeeper-\*.jar,guava-\*.jar in hive-\*/lib from hbase-\*/lib
- Copy the all the jars from hbase-\*/lib folder to HADOOP\_HOME/lib folder
- bin/hive
- *set hbase.zookeeper.quorum=localhost;*
- *create external TABLE hbase\_table(key int, value string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,data:name") TBLPROPERTIES ("hbase.table.name" = "testtable");*
- *insert overwrite table hbase\_table select pid, pname from patient;*
- *select \* from hbase\_table;*

**ThAnK yOu**

