Apache Hive

CMM705 - Big Data Programming
Lecture 6

Overview

- Hive
  - Background of Hive
  - Hive vs Pig
  - Hive Architecture
  - Limitation of Hive
  - Data types
  - Data Models
  - Partitioning and bucketing
- Lab session on Hive
  - Setting up Hadoop on single node
  - Setting up Hive
  - Running Hive

Background of Hive

- Started at Facebook
- Data was collected into Oracle DB by nightly cron jobs
- Grew from 10 of GBs in 2006 to 1 TB/day in 2007 and now it's 10x higher
    - > 950 Million Users
    - > 500 TB per day
    - > 70k queries per day
    - > 300m photos per day
- Users (employees) know SQL well

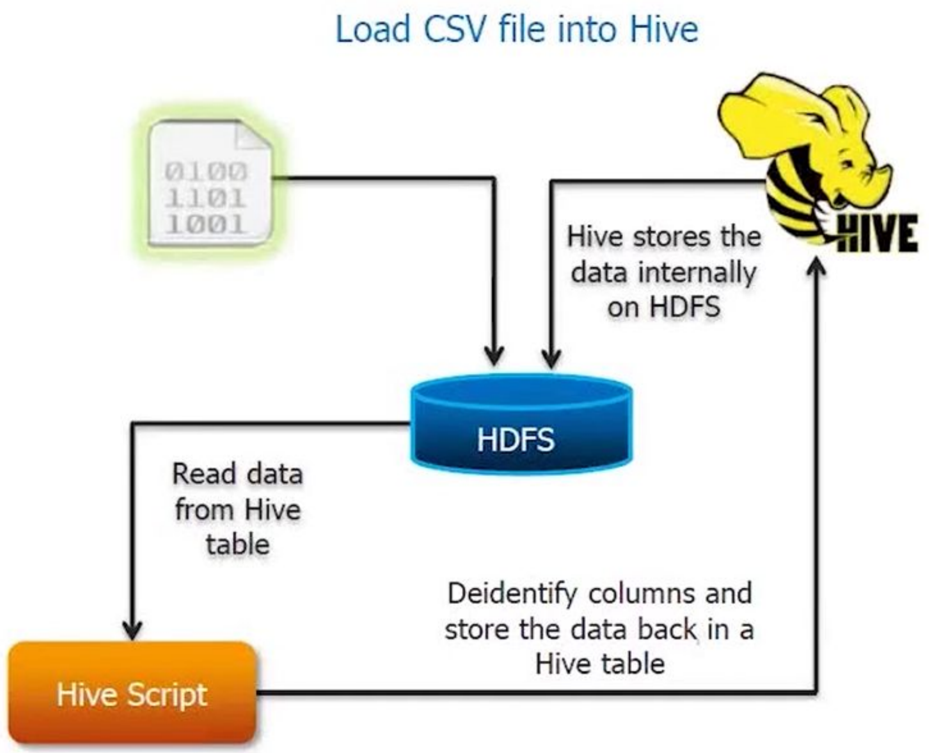
What is Hive ?

- A data warehousing package built on top of Hadoop
- Can be used for data analytics
- Targeted on users who are familiar with SQL
- Called as HiveQL
- No need to learn Java and Hadoop API :)
- Cannot process semi structured data.
    - Additional parsing (transformations) or schema inference needed
    - process semi-structured data by using UDFs (User Defined Functions) or by transforming data into a structured format
- Enable easy data ETL
    - Built-in Functions and Libraries

Hive

- Partitioning based on column values and Bucketing further segments to manageable chunks
- Schema flexibility (Schema on read) and evolution (Changing schema)
- Easy to plug-in custom mapreduce code
- JDBC/ODBC drivers are available
- HIVE tables can be defined directly on HDFS
- Extensible types, formats, functions & scripts
- Write once read many times (store years worth of data and analyze)
- Not a RDBMS or a database, just resembling a RDBMS for convenience

Hive (Managed Table)


Load CSV file into Hive
0100
1101
1001
HIVE
Hive stores the data internally on HDFS
HDFS
Read data from Hive table
Deidentify columns and store the data back in a Hive table
Hive Script

Where to use Hive

- Log Processing
  - identify patterns, relationships, or trends within your data
- Customer facing BI - Top 10 users, etc
  - use ORDER BY or LIMIT clauses
- Data mining
  - identify patterns, relationships, or trends within your data
- Document Indexing
  - By creating tables that map to your document structure and indexing the table
- Predictive Modeling, Hypothesis Testing
  - Prepare and aggregate the data needed for your models
  - Exported to machine learning frameworks

Hive vs Pig ?

Pig

- Developed by Yahoo
- Procedural data flow

```
LOAD 'input_data' USING PigStorage(',')
AS (field1:chararray, field2:int);
FILTER A BY field2 > 100;
GROUP B BY field1;
FOREACH C GENERATE group, COUNT(B);
```

- Mostly used by Programmers and Researchers
- Implicit Schemas

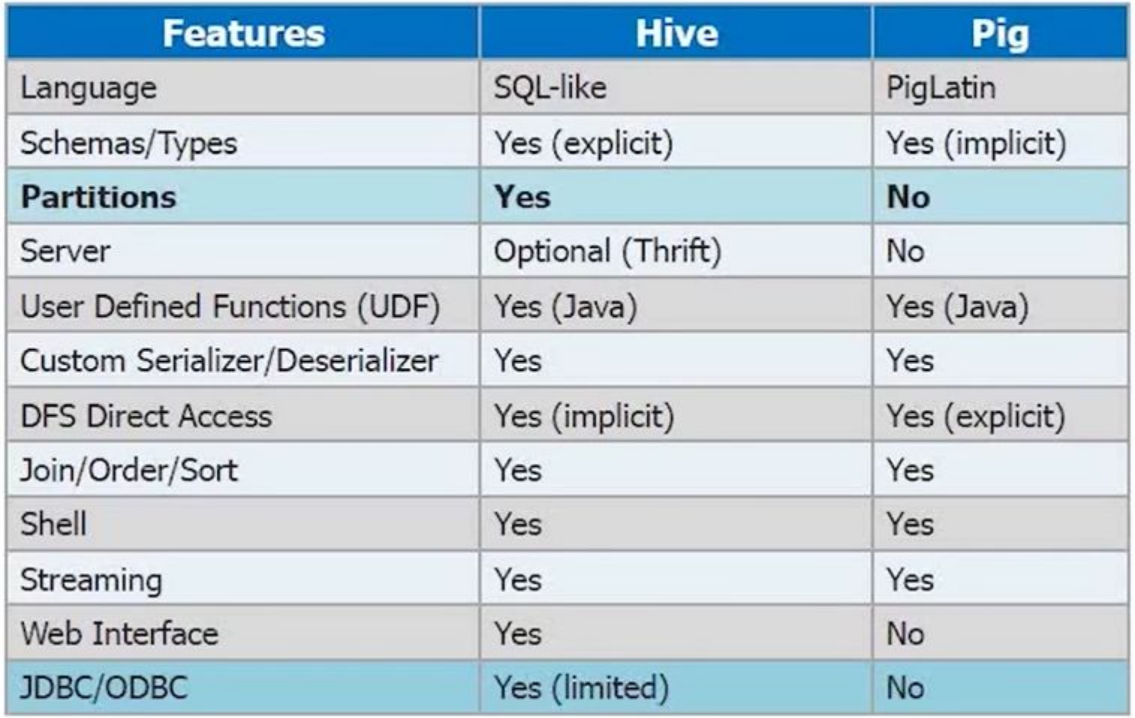
Hive

- Developed by Facebook
- Declarative SQL

```
SELECT field1, COUNT(*) FROM input_data
WHERE field2 > 100 GROUP BY field1;
```

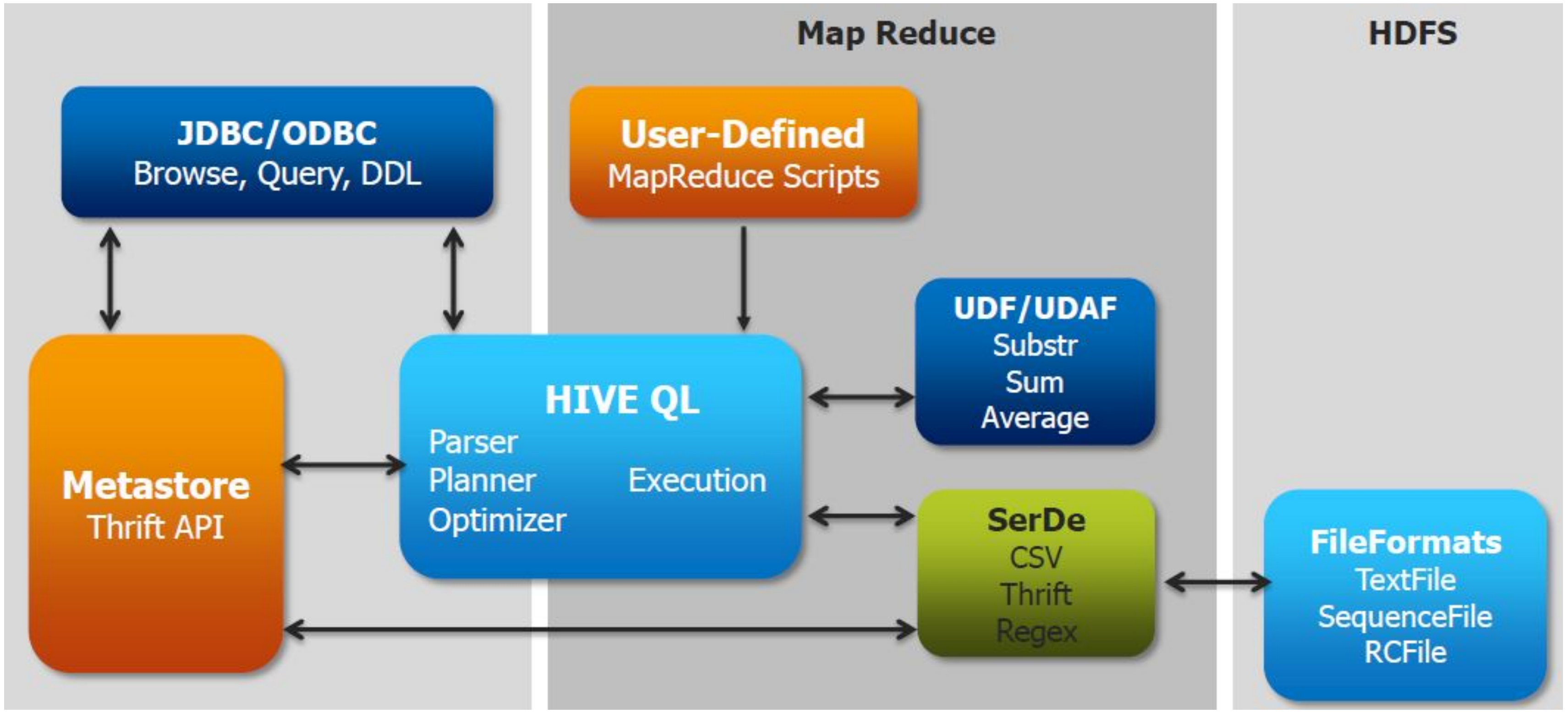
- Mostly used by Analysts generating reports daily
- Provided partitions

Hive vs Pig ? ...

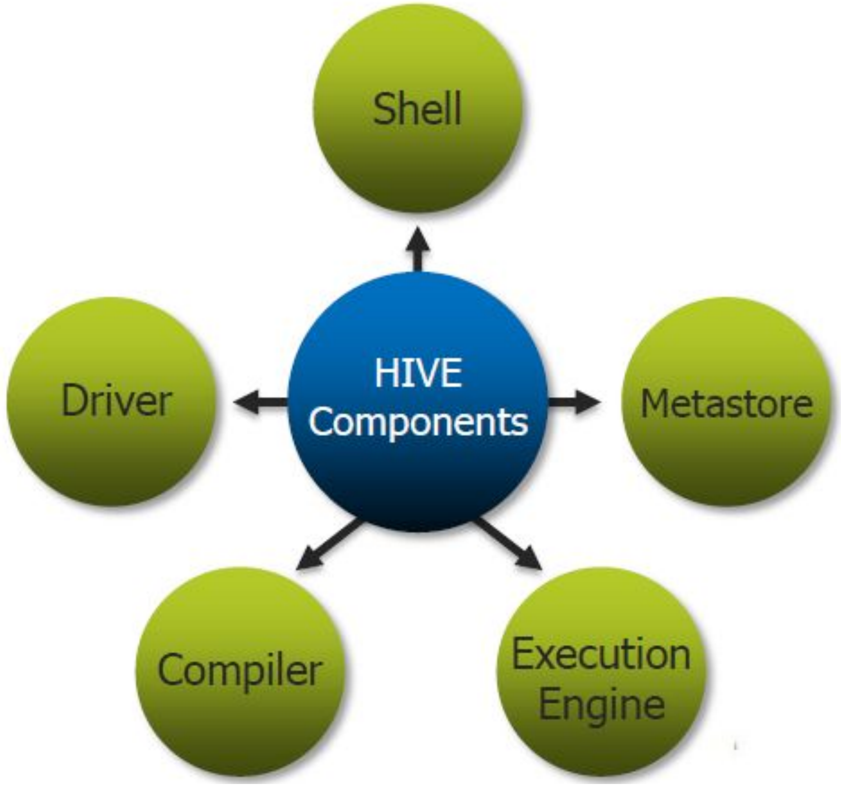


Features	Hive	Pig
Language	SQL-like	PigLatin
Schemas/Types	Yes (explicit)	Yes (implicit)
Partitions	**Yes**	**No**
Server	Optional (Thrift)	No
User Defined Functions (UDF)	Yes (Java)	Yes (Java)
Custom Serializer/Deserializer	Yes	Yes
DFS Direct Access	Yes (implicit)	Yes (explicit)
Join/Order/Sort	Yes	Yes
Shell	Yes	Yes
Streaming	Yes	Yes
Web Interface	Yes	No
JDBC/ODBC	Yes (limited)	No

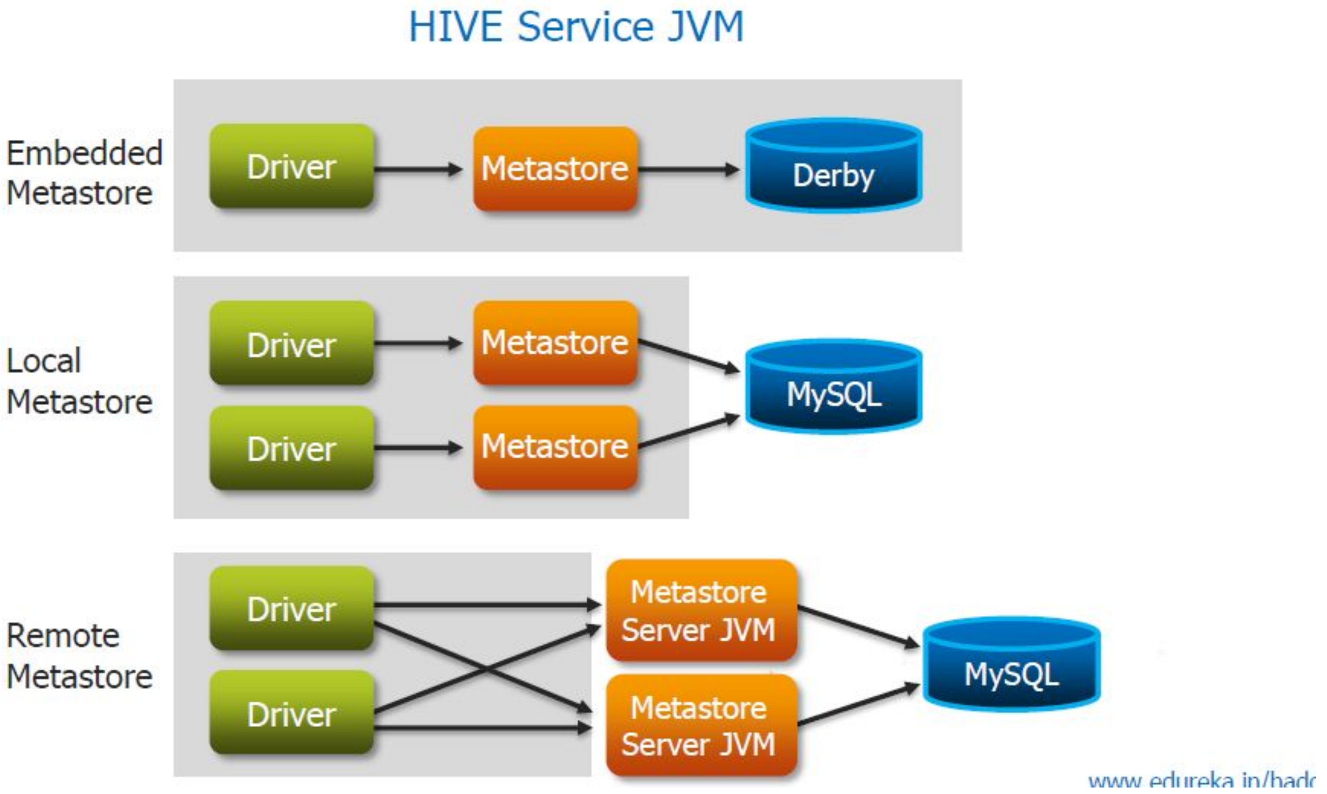
Hive Architecture



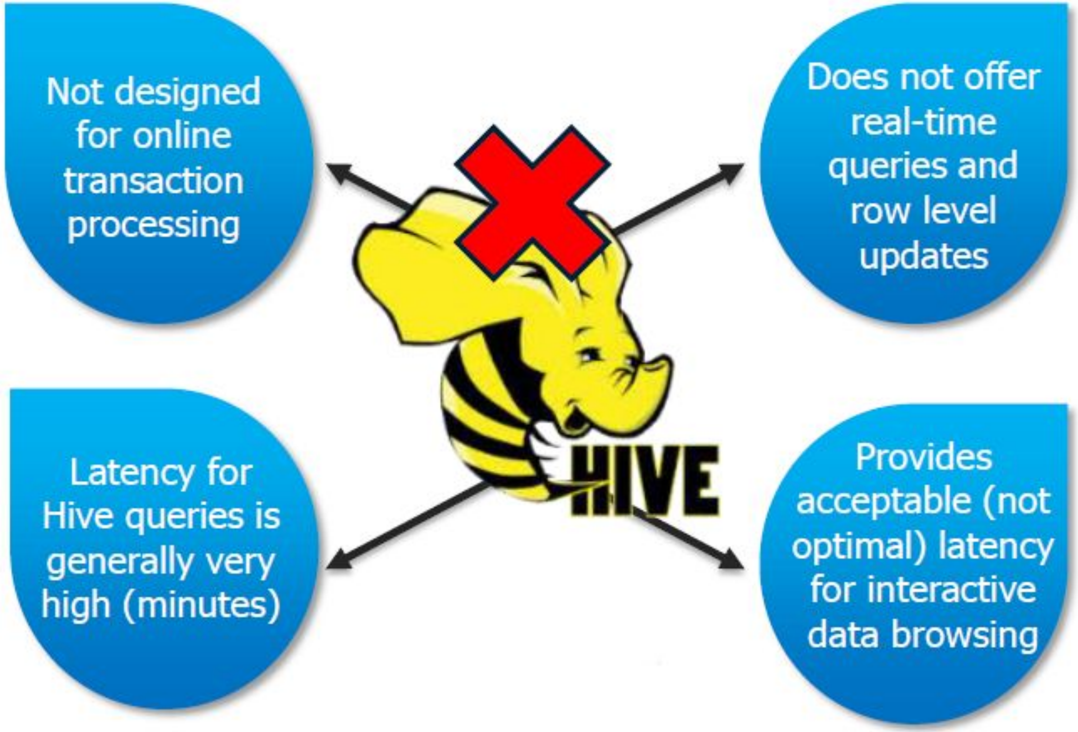
Components of Hive



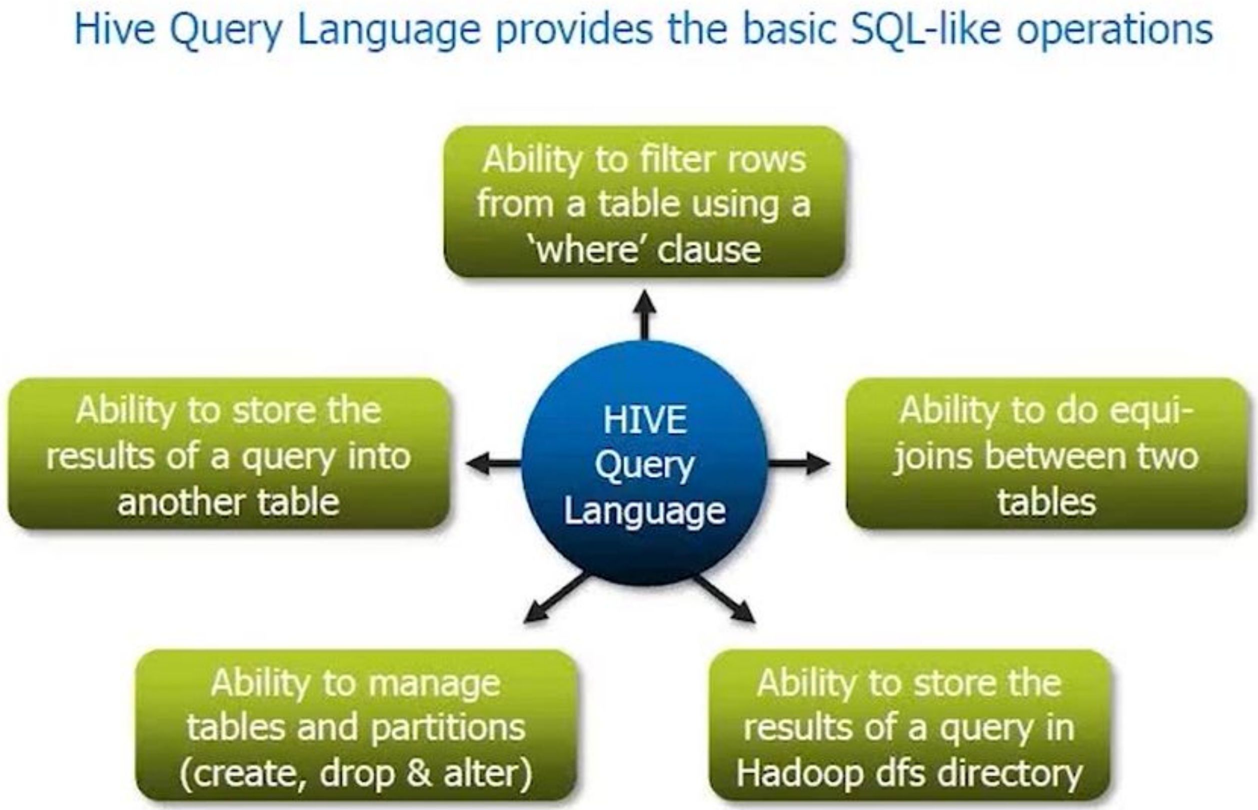
Metastore


HIVE Service JVM
Embedded Metastore
Driver
Metastore
Derby
Local Metastore
Driver
Metastore
Driver
Metastore
MySQL
Remote Metastore
Driver
Driver
Metastore Server JVM
Metastore Server JVM
MySQL
www.edureka.in/hadc

Limitation of Hive


Not designed for online transaction processing
Does not offer real-time queries and row level updates
HIVE
Latency for Hive queries is generally very high (minutes)
Provides acceptable (not optimal) latency for interactive data browsing

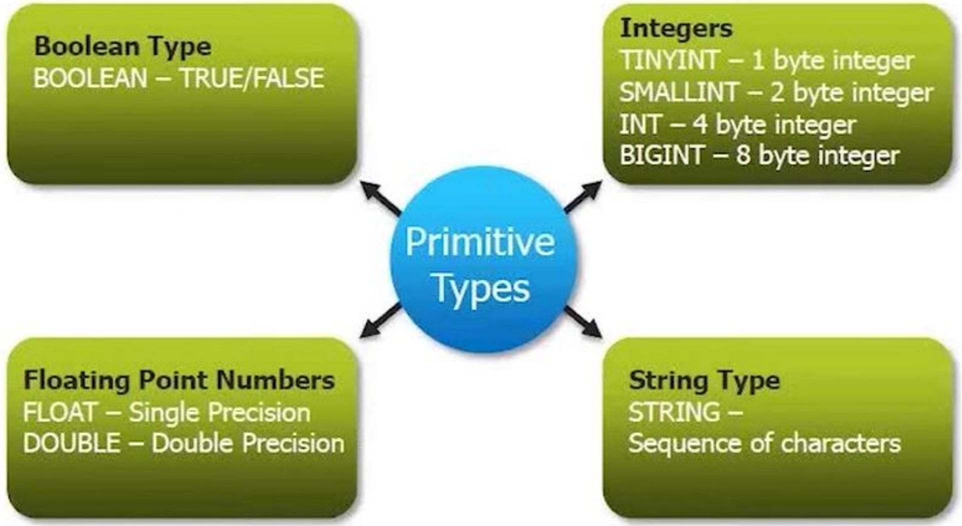
Ability of HiveQL


Hive Query Language provides the basic SQL-like operations
HIVE Query Language
Ability to filter rows from a table using a 'where' clause
Ability to store the results of a query into another table
Ability to do equi-joins between two tables
Ability to manage tables and partitions (create, drop & alter)
Ability to store the results of a query in Hadoop dfs directory

Data Handling in Hive

- Hive does not verify the data when it's loaded.
- It only verify the data when query is issued - Schema on Read
- No updates and transactions
  - Hive tables are generally append-only; once data is written to a table, you cannot update individual rows or cells within that table as you would in a traditional database
  - Did not support ACID (Atomicity, Consistency, Isolation, Durability) transactions
  - From Hive 0.14, Hive introduced ACID transactions with **limited transaction support**, specifically for insert, update, and delete operations

Data Types Supported



+

Array, Union

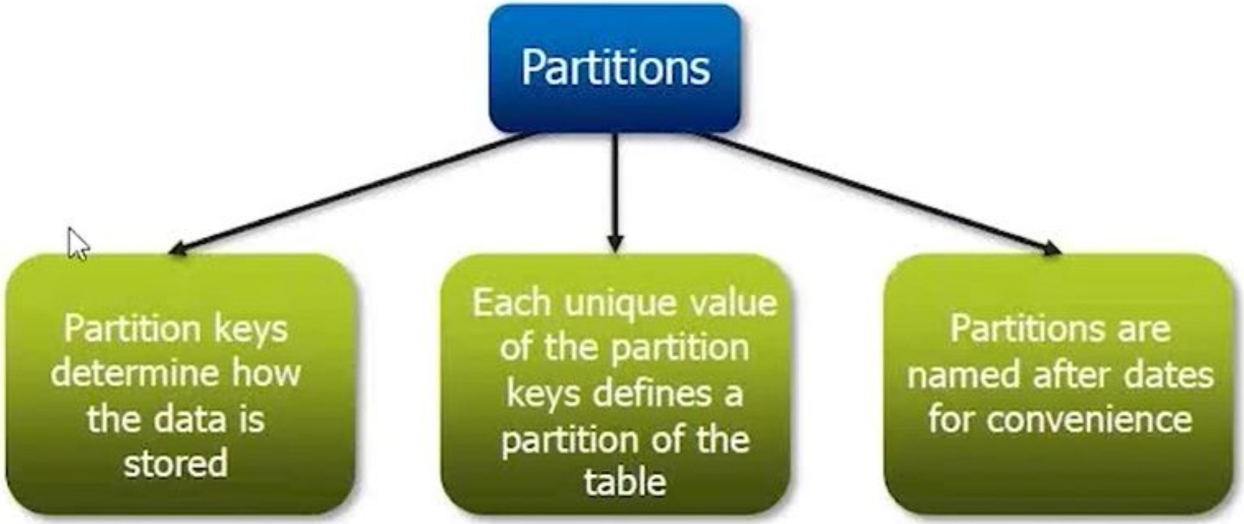
Struct, Map

Hive Data Models

- Database
  - Namespaces
- Tables
  - Schemas in namespaces
- Partitions
  - How data is stored in HDFS
  - Grouping data based on some column
- Buckets or Clusters
  - Partitions divided further into buckets based on some other column
  - Used for data sampling

Partitioning

Partition means dividing a table into a coarse grained parts based on the value of a partition column such as a date. This make it faster to do queries on slices of the data.


Partitions
Partition keys determine how the data is stored
Each unique value of the partition keys defines a partition of the table
Partitions are named after dates for convenience

Lab Session

Starting the Docker Image

- Clone the below git repository
```
git clone https://github.com/ramindu-msc/iit
git pull origin main
```
```
Or if you have the repository created already
ccd lab5/hadoop-hive-dockercompose/
```

- Change mapreduce-design-intro/hadoop-dockercompose/docker-compose.yaml’s /home/iitgcpuser/iit/lab5/hadoop-hive-dockercompose/resources to your repository cloned path

- And start the containers with the following command(also by replacing the path)
```
sudo docker compose -f
/home/iitgcpuser/iit/lab5/hadoop-hive-dockercompose/docker-compose-new.yaml up -d
```

- Check the datanode is connected to namenode and fully started
```
sudo docker logs -f hadoop-hive-dockercompose-datanode-1
```

Running Hive Commands

- Incase of an error as follows

```
✘ Container hive-metastore-init                service "hive-metastore-init" didn't
complete successfully: exit 1
```

- Run the following commands to get to hive shell and create database

```
sudo docker compose -f
/home/iitgcpuser/iit/lab5/hadoop-hive-dockercompose/docker-compose.yaml down -v

 sudo docker volume rm $(sudo docker volume ls -q)
```

- Restart the containers

```
sudo docker compose -f
/home/iitgcpuser/iit/lab5/hadoop-hive-dockercompose/docker-compose.yaml up -d
```

Running Hive Commands

- Navigate to Run the docker image and run the following commands

  ```
  sudo docker exec -it hive-server bash
  ```

- Navigate to Run the docker image and run the following commands

  ```
  hive
  create database telecom;
  show databases;
  create database telecom_backup comment 'holds backup data';
  describe database extended telecom_backup;
  ```

Creating a Managed Table

- Open a separate terminal for namenode

```
docker exec -it namenode bash
hdfs dfs -ls /user/hive/warehouse/telecom.db/recharge
```

- Run the following command to create a table in hive shell

```
use telecom;
create table recharge( cell_no int, city string, name string, price float) row format
delimited fields terminated by ',' ;
describe extended recharge;
```

- In the terminal for namenode

```
hdfs dfs -ls /user/hive/warehouse/telecom.db/recharge
```

Creating a Managed Table..

- Run the following commands insert data, select data

```
INSERT INTO recharge (cell_no,city,name,price) VALUES (999090,"sl","fernando",30.0);
```

```
SELECT * FROM recharge;
```

- Load data from local disk

```
LOAD DATA LOCAL INPATH '/opt/recharge2.input' INTO TABLE recharge;
```

```
SELECT * FROM recharge;
```

- Run the following commands in namenode terminal

```
hdfs dfs -ls /user/hive/warehouse/telecom.db/recharge
```

```
hdfs dfs -cat ____________
```

Creating a External Table

- In the terminal for namenode

```
hdfs dfs -mkdir -p /user/hive/warehouse/telecom.db/recharge2

hdfs dfs -put /opt/recharge2.input  /user/hive/warehouse/telecom.db/recharge2

hdfs dfs -cat /user/hive/warehouse/telecom.db/recharge2/recharge2.input
```

- Run the following commands in namenode shell

```
create external table recharge_external(cell_no int, city string, name string, price float)

row format delimited fields terminated by ','  LOCATION

'hdfs://namenode:8020/user/hive/warehouse/telecom.db/recharge2';


SELECT * FROM recharge_external;
```

Creating a External Table..

- In the terminal for namenode

```
echo -e "\n11436,sl,de silva,100" | hdfs dfs -appendToFile -
/user/hive/warehouse/telecom.db/recharge2/recharge2.input
```

- In the terminal for namenode

```
echo "11436,sl,ramindu,100" > new_data.txt
hdfs dfs -put -f new_data.txt /user/hive/warehouse/telecom.db/recharge2/recharge2.input
```

- Run the following commands in namenode shell

```
SELECT * FROM recharge_external;
```

Partitioning and Bucketing

```
use telecom;
create table rechargeP (
     cell_no int,
     name string,
     price float)
partitioned by (city string)
clustered by (name) into 10 buckets
     row format delimited fields terminated by ','
          stored as textfile;
```

- Same command in single line

```
create table rechargeP ( cell_no int, name string, price float) partitioned by (city string)
clustered by (name) into 10 buckets row format delimited fields terminated by ',' stored as
textfile; LOCATION 'hdfs://namenode:8020/user/hive/warehouse/telecom.db/partioned';
```

Partitioning and Bucketing …

- Set properties

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

- Load data to the partitioned table from recharge_external:

```
INSERT OVERWRITE TABLE rechargeP PARTITION (city) SELECT r2.cell_no, r2.name,
r2.price, r2.city FROM recharge_external r2;
```

- See the partitions

```
SHOW PARTITIONS rechargeP;
```

Writing to Local Files

- Insert into local file

```
INSERT OVERWRITE  LOCAL DIRECTORY '/temp/h_result'  ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' select * from recharge;

Exit;
```

- Check the local file

```
cat /temp/h_result/000000_0
```

What we covered

- Hive
  - Background of Hive
  - Hive vs Pig
  - Hive Architecture
  - Limitation of Hive
  - Data types
  - Data Models
  - Partitioning and bucketing
- Lab session on Hive
  - Setting up Hadoop on single node
  - Setting up Hive
  - Running Hive

Resources

- http://www.edureka.co/blog/hive-data-models/?utm_source=youtube&utm_medium=referral&utm_campaign=hive-tut1
- http://www.edureka.co/blog/pig-vs-hive/?utm_source=youtube&utm_medium=referral&utm_campaign=hive-tut1
- https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStarted-InstallationandConfiguration
- https://www.youtube.com/watch?v=MoKW5eY5yVY
- https://www.youtube.com/watch?v=tKNGB5IZPFE
- http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/SingleCluster.html