BITS Pilani presentation

BITS Pilani
Pilani Campus

Dr. Vivek V. Jog
Dept. of Computer Engineering

innovate achieve lead

BITS Pilani

Pilani Campus

Big Data Systems (S1-24_CCZG522)

Lecture No.9

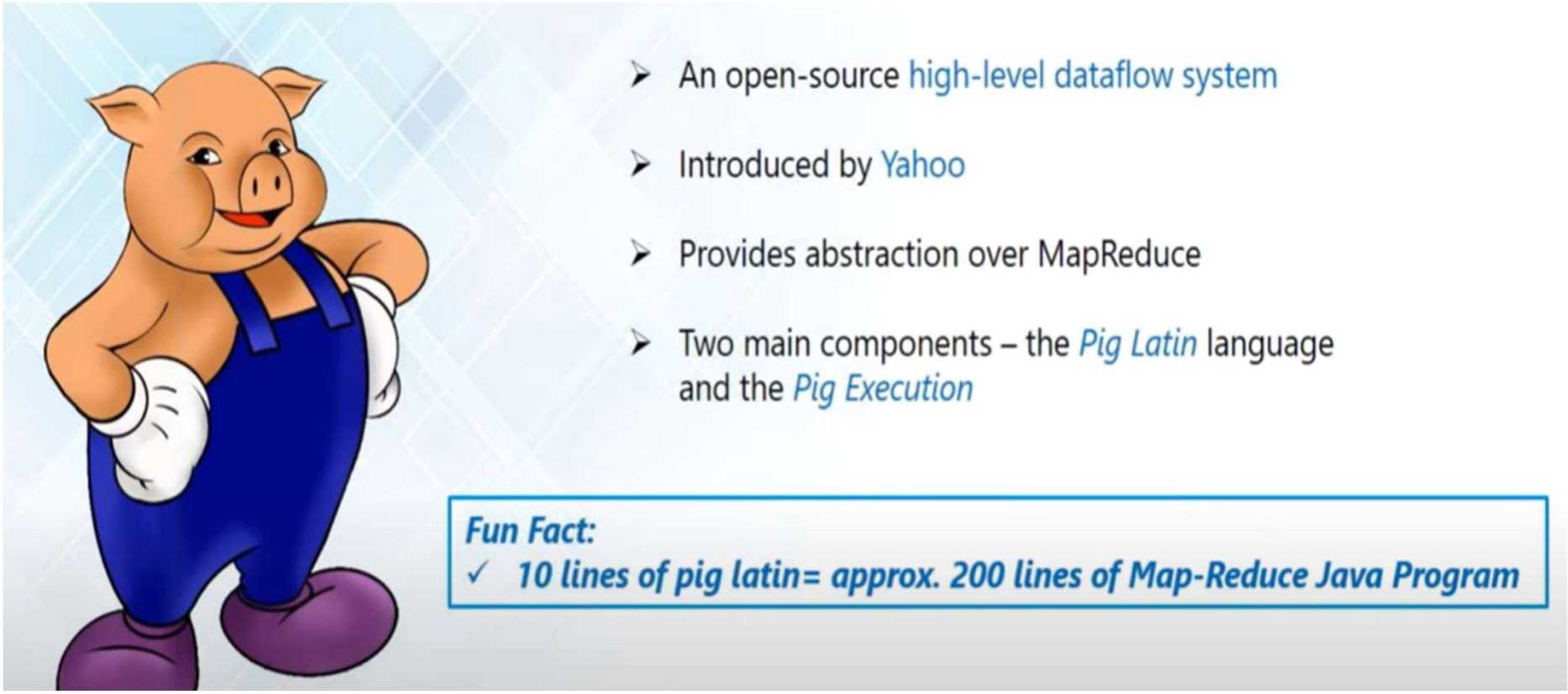
PIG

- Entry of Apache Pig
- Pig vs MapReduce
- Twitter Case Study on Apache Pig
- Apache Pig Architecture
- Pig Components
- Pig Data Model & Operators
- Running Pig Commands and Pig Scripts (Log Analysis)

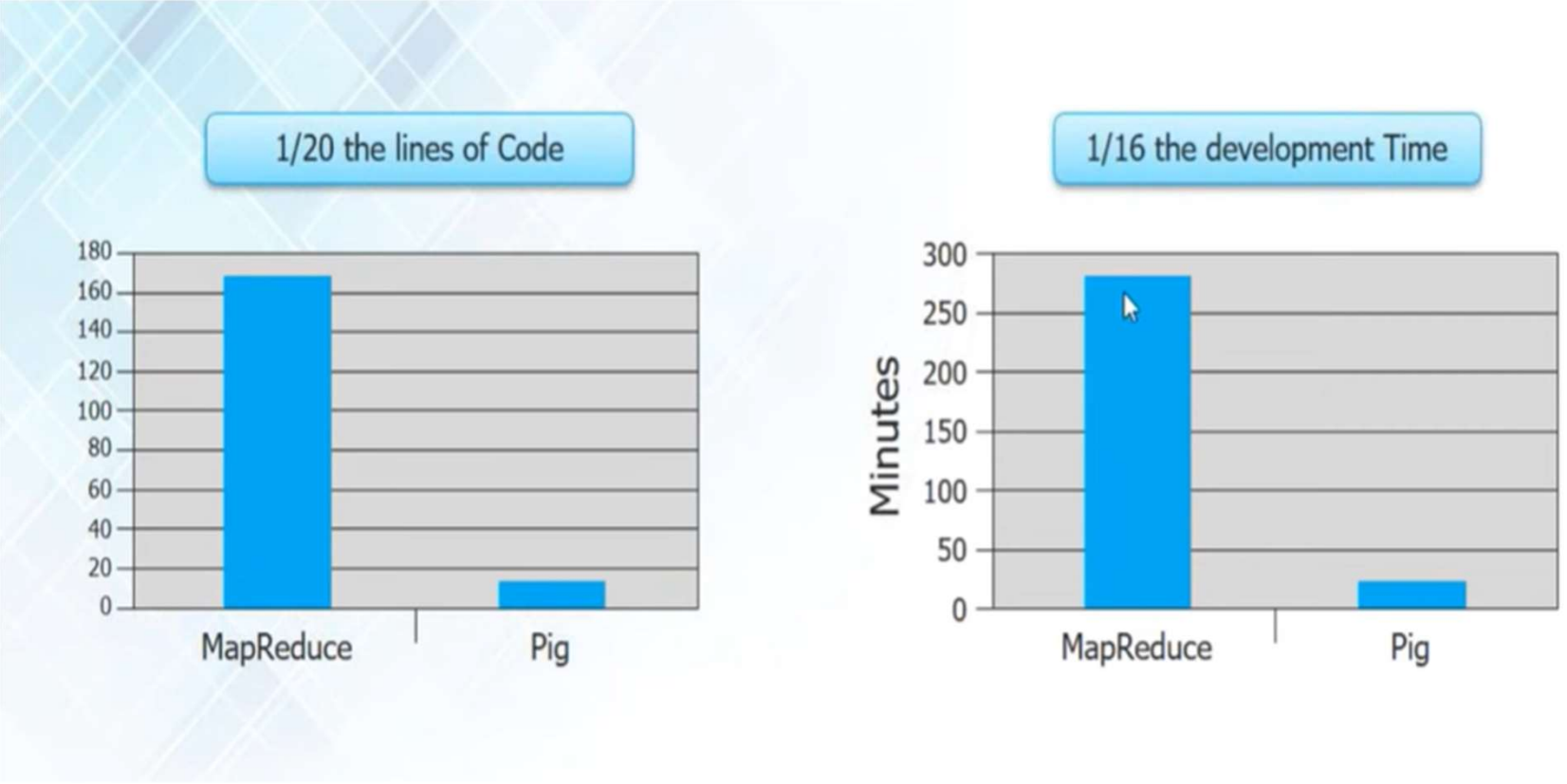
No Need to be a Programmer

In MapReduce, you need to write a program in Java/Python to process the data.

Developed by YAHOO



Clear Winner

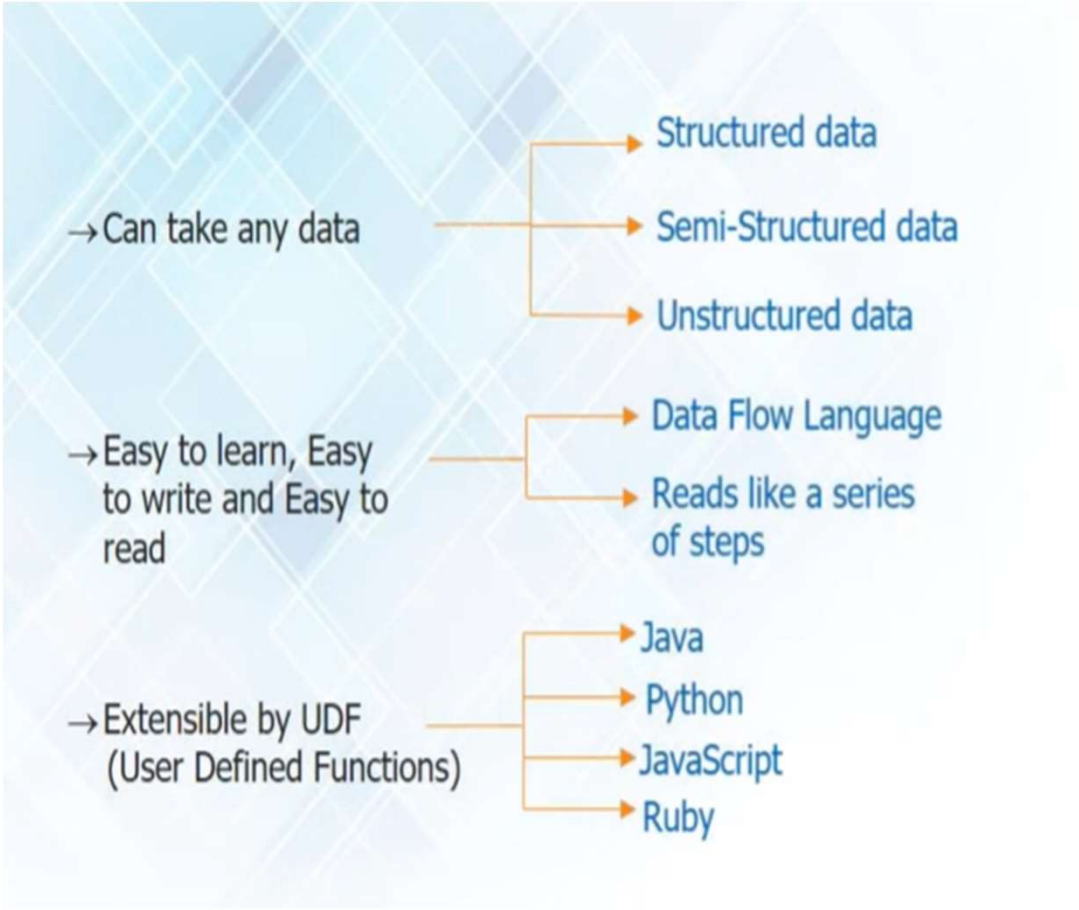


Pig v/s MapReduce

- High-level data flow tool
- No need to write complex programs
- Built-in support for data operations like joins, filters, ordering, sorting etc.
- Provides nested data types like tuples, bags, and maps

- Low-level data processing paradigm
- You need write programs in Java/Python etc.
- Performing data operations in MapReduce is a humongous task
- Nested data types are not there in MapReduce

Pig Specific



→Provides common data operations filters, joins, ordering, etc. and nested data types tuples, bags, and maps missing from MapReduce.

→An ad-hoc way of creating and executing map-reduce jobs on very large data sets

→Open source and actively supported by a community of developers.

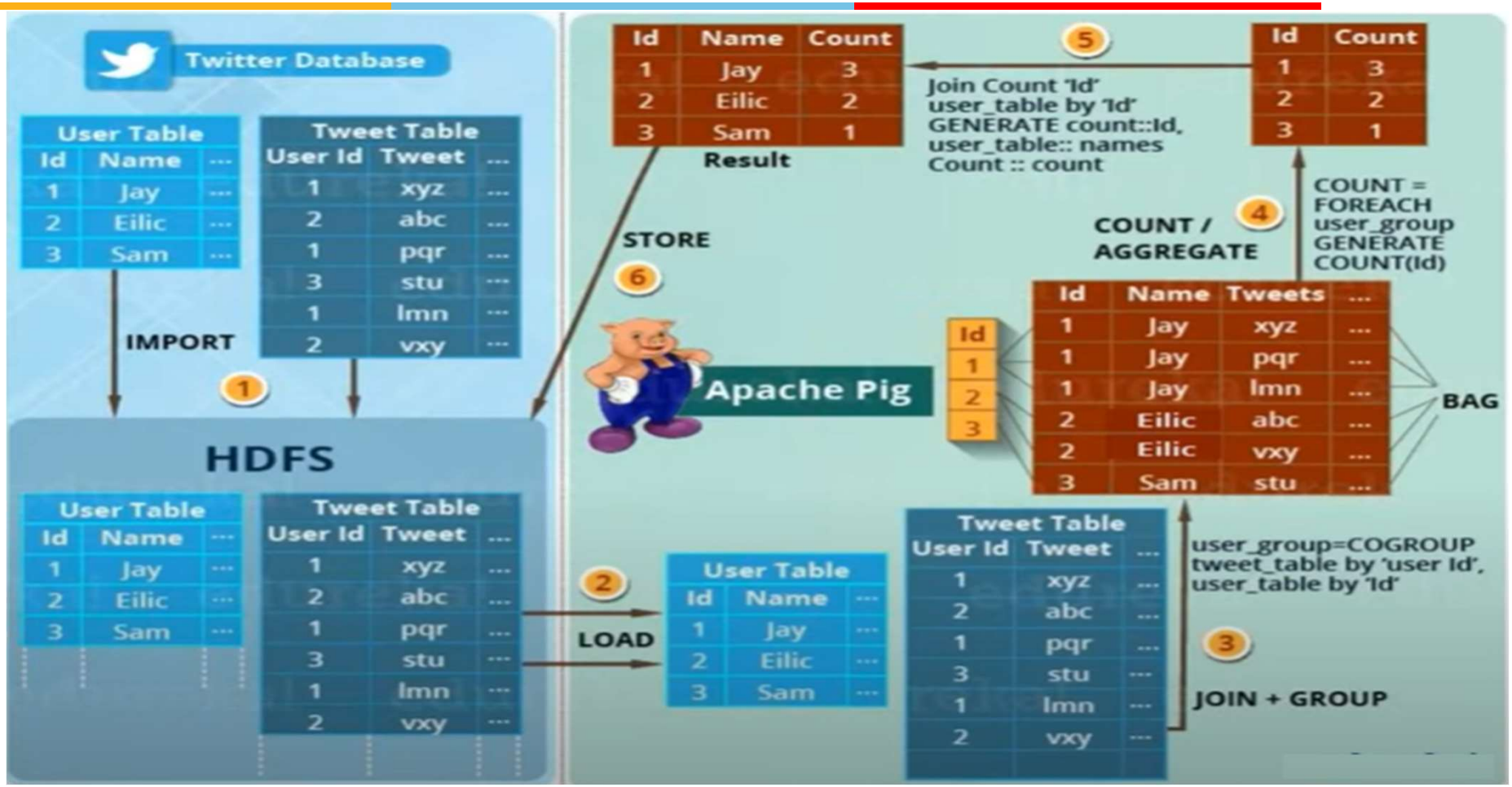
Pig – Hello World

- Twitter's data was growing at an accelerating rate (i.e. 10 TB/day).
- Thus, Twitter decided to move the archived data to HDFS and adopt Hadoop for extracting the business values out of it.
- Their major aim was to analyse data stored in Hadoop to come up with the multiple insights on a daily, weekly or monthly basis.

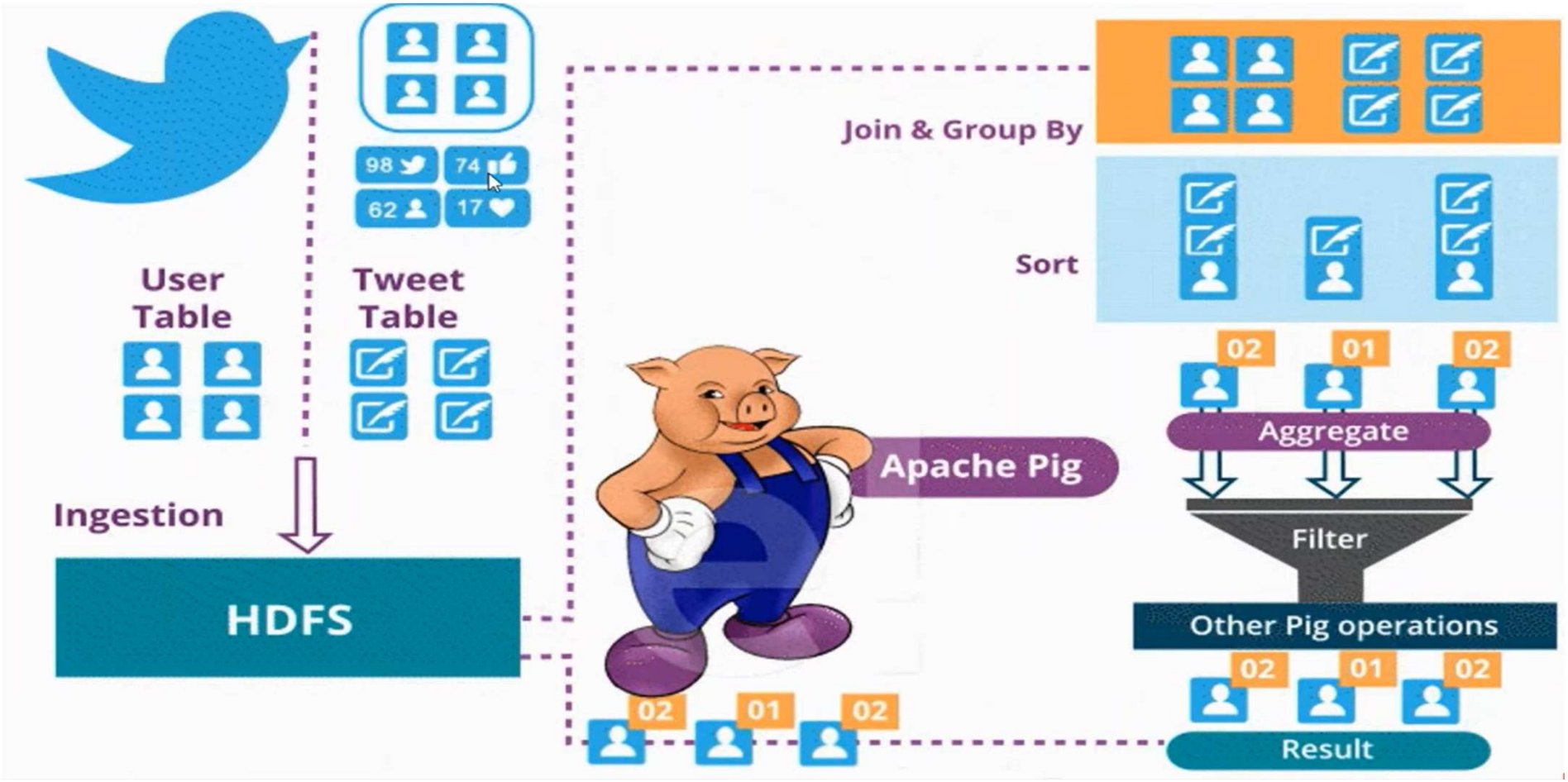
Let me talk about one of the insight they wanted to know.

Analyzing how many tweets are stored per user, in the given tweet tables?

Working


Twitter Database
User Table
Id Name ...
1 Jay ...
2 Eilic ...
3 Sam ...
Tweet Table
User Id Tweet ...
1 xyz ...
2 abc ...
1 pqr ...
3 stu ...
1 lmn ...
2 vxy ...
IMPORT
1
HDFS
User Table
Id Name ...
1 Jay ...
2 Eilic ...
3 Sam ...
Tweet Table
User Id Tweet ...
1 xyz ...
2 abc ...
1 pqr ...
3 stu ...
1 lmn ...
2 vxy ...
2
LOAD
User Table
Id Name ...
1 Jay ...
2 Eilic ...
3 Sam ...
Apache Pig
STORE
6
Id Name Count
1 Jay 3
2 Eilic 2
3 Sam 1
Result
5
Join Count 'Id'
user_table by 'Id'
GENERATE count::Id,
user_table:: names
Count :: count
Id Count
1 3
2 2
3 1
COUNT / AGGREGATE
4
COUNT =
FOREACH
user_group
GENERATE
COUNT(Id)
Id
1
2
3
Id Name Tweets ...
1 Jay xyz ...
1 Jay pqr ...
1 Jay lmn ...
2 Eilic abc ...
2 Eilic vxy ...
3 Sam stu ...
BAG
Tweet Table
User Id Tweet ...
1 xyz ...
2 abc ...
1 pqr ...
3 stu ...
1 lmn ...
2 vxy ...
user_group=COGROUP
tweet_table by 'user Id',
user_table by 'Id'
3
JOIN + GROUP

Overview


98
74
62
17
User Table
Tweet Table
Ingestion
HDFS
Join & Group By
Sort
Apache Pig
02
01
02
Aggregate
Filter
Other Pig operations
02
01
02
Result

Architecture

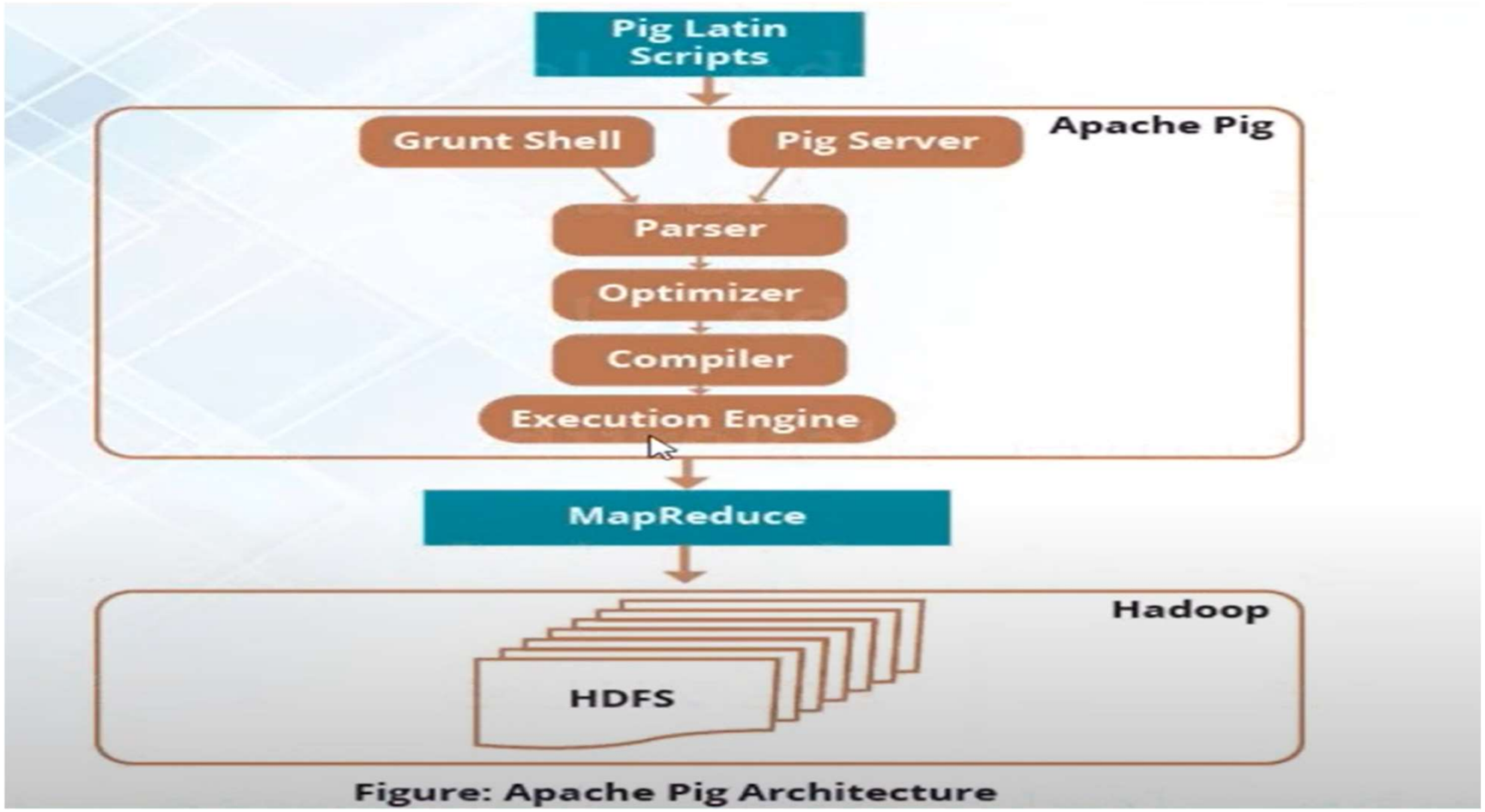
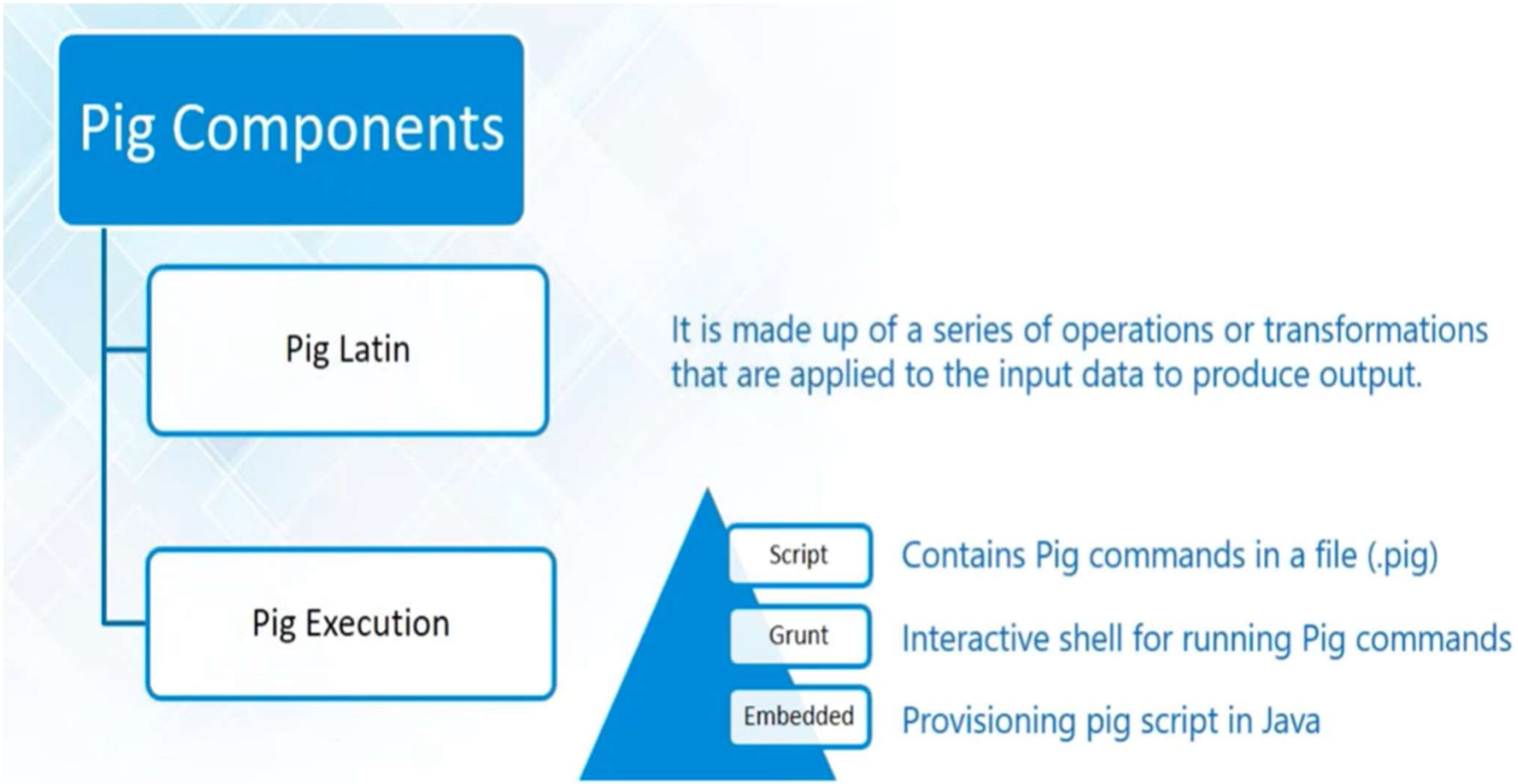


Figure: Apache Pig Architecture

Components


Pig Components
Pig Latin
Pig Execution
It is made up of a series of operations or transformations that are applied to the input data to produce output.
Script
Contains Pig commands in a file (.pig)
Grunt
Interactive shell for running Pig commands
Embedded
Provisioning pig script in Java

Modes of Execution

You can run Apache Pig in 2 modes:

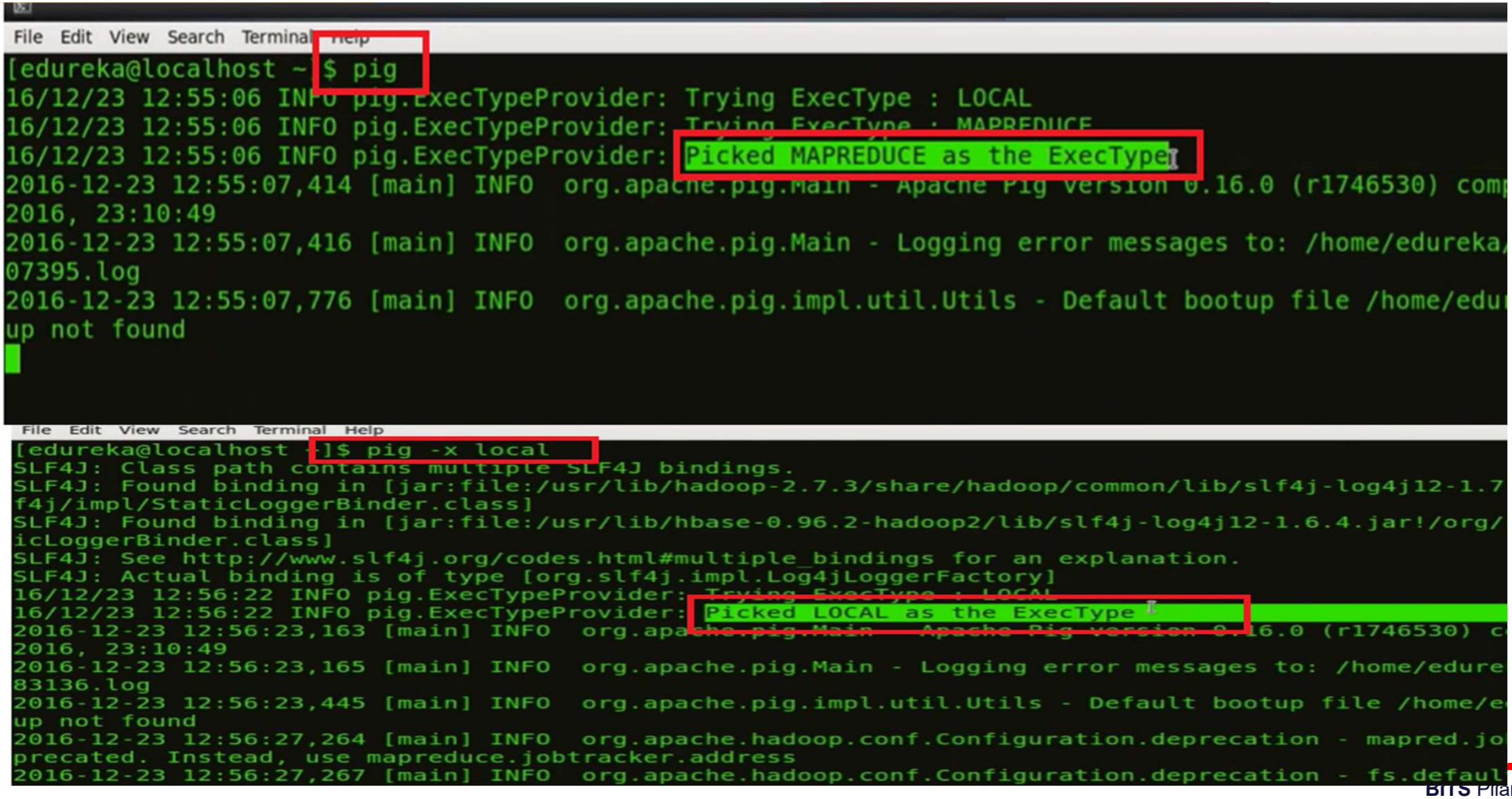
MapReduce Mode – This is the default mode, which requires access to a Hadoop cluster and HDFS installation. The input and output in this mode are present on HDFS.

Command: pig

Local Mode – With access to a single machine, all files are installed and run using a local host and file system. Here the local mode is specified using '-x flag' (pig -x local). The input and output in this mode are present on local file system.

Command: pig –x local

MapReduce & Local Mode



Data Models

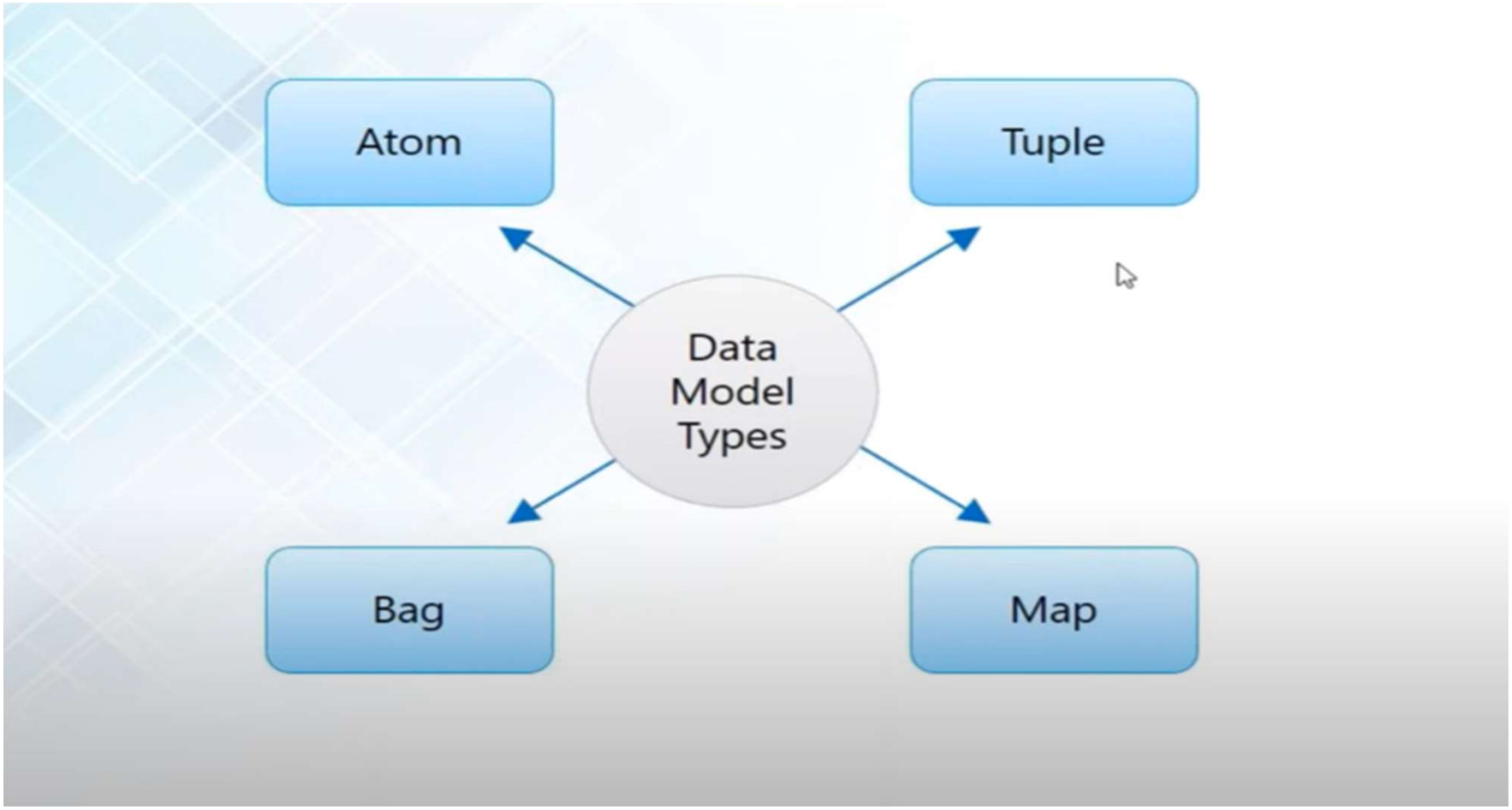


Figure: Apache Pig Data Model

edureka!

- Tuple is an ordered set of fields which may contain different data types for each field.

  Example of tuple – (1, Linkin Park, 7, California)

- A Bag is a collection of a set of tuples and these tuples are subset of rows or entire rows of a table.

  Example of a bag – {(Linkin Park, 7, California), (Metallica, 8), (Mega Death, Los Angeles)}

Map & Atom

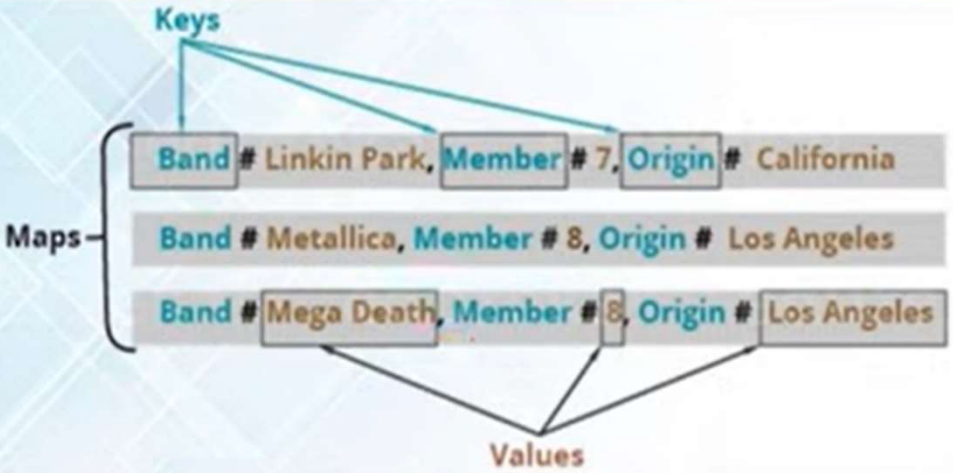


Figure: Map Example

- A Map is key-value pairs used to represent data elements.

  Example of maps– [band#Linkin Park, members#7], [band#Metallica, members#8]

- Atoms are basic data types which are used in all the languages like string, int, float, long, double, char[], byte[]

Operators for DB Operatrions

Operator	Description
LOAD	Load data from the local file system or HDFS storage into Pig
FOREACH	Generates data transformations based on columns of data
FILTER	Selects tuples from a relation based on a condition
JOIN	Join the relations based on the column
ORDER BY	Sort a relation based on one or more fields
STORE	Save results to the local file system or HDFS
DISTINCT	Removes duplicate tuples in a relation
GROUP	Groups together the tuples with the same group key (key field)
COGROUP	It is same as GROUP. But COGROUP is used when multiple relations re involved

move data to HDFS

```
edureka@localhost:~
File  Edit  View  Search  Terminal  Help
[edureka@localhost ~]$ hadoop dfs -put Desktop/Datasets/input /pigInput
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/12/23 13:16:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
iltin-java classes where applicable
[edureka@localhost ~]$
```

Sample Dtata

input

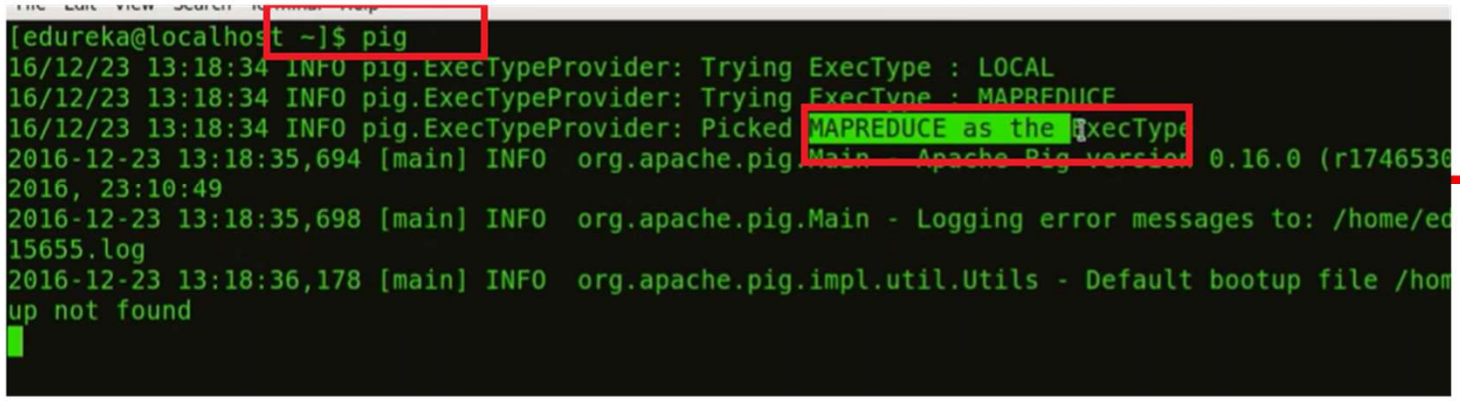
```
111,John,Sales,Austin
222,Alex,Marketing,New York
333,Philip,Operation,Sacramento
444,Terry,Sales,New York
555,Jessi,Development,Boston
```

Data loaded in HDFS

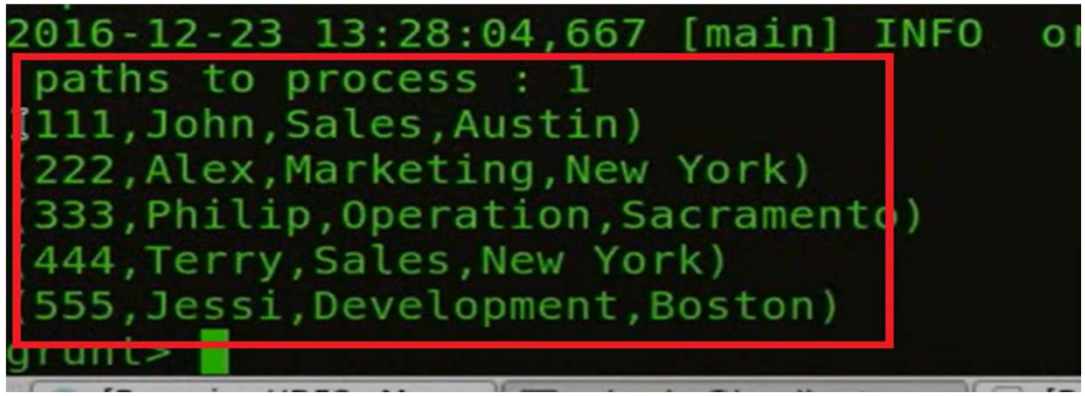
```
iltin-java classes where applicable
[edureka@localhost ~]$ hadoop dfs -cat /pigInput
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/12/23 13:17:23 WARN util.NativeCodeLoader: Unable to load native-hadoop
iltin-java classes where applicable
111,John,Sales,Austin
222,Alex,Marketing,New York
333,Philip,Operation,Sacramento
444,Terry,Sales,New York
555,Jessi,Development,Boston
[edureka@localhost ~]$
```

Execute Pig in MapReduce mode



Load data from HDFS to Pig (you can also load data without specifying schema)



Use of "foreach"

```
(222,Alex,Marketing,New York)
(333,Philip,Operation,Sacramento)
(444,Terry,Sales,New York)
(555,Jessi,Development,Boston)
grunt> emp_foreach = foreach employee generate name,department;
```

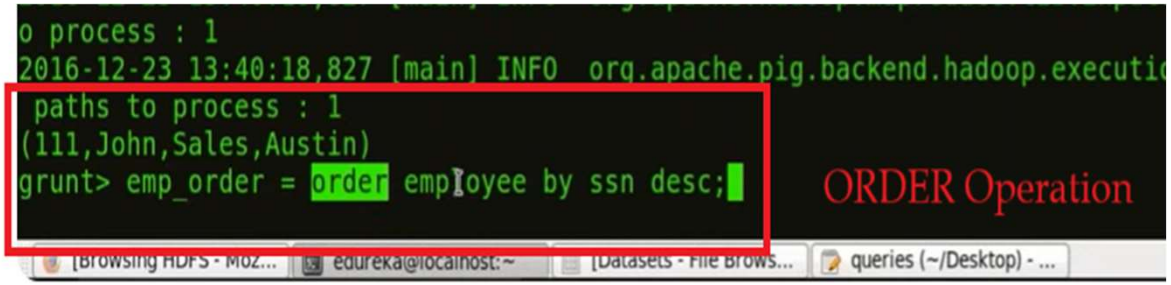
Selecting two column from every ROW

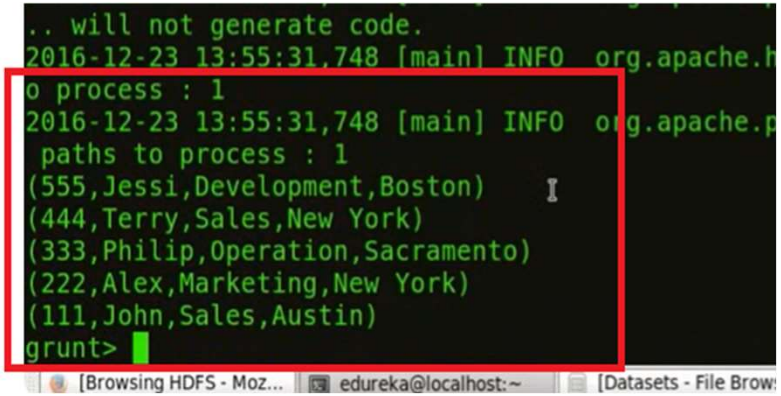
```
2016-12-23 13:35:23,637 [main] INFO  org.apache.pig.
 paths to process : 1
(John,Sales)
(Alex,Marketing)
(Philip,Operation)
(Terry,Sales)
(Jessi,Development)
grunt>
```

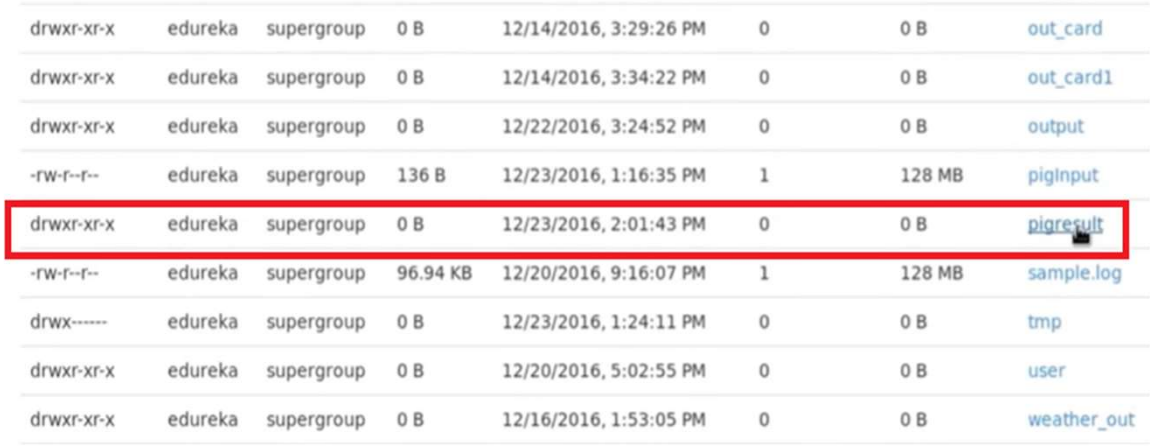
Filter Employee from Austin -- Filter operation

```
(Terry,Sales)
(Jessi,Development)
grunt> emp_filter = filter employee by city == 'Austin';
```

```
.. will not generate code.
2016-12-23 13:40:18,827 [main] INFO  org.apache
o process : 1
2016-12-23 13:40:18,827 [main] INFO  org.apache
 paths to process : 1
(111,John,Sales,Austin)
grunt>
```





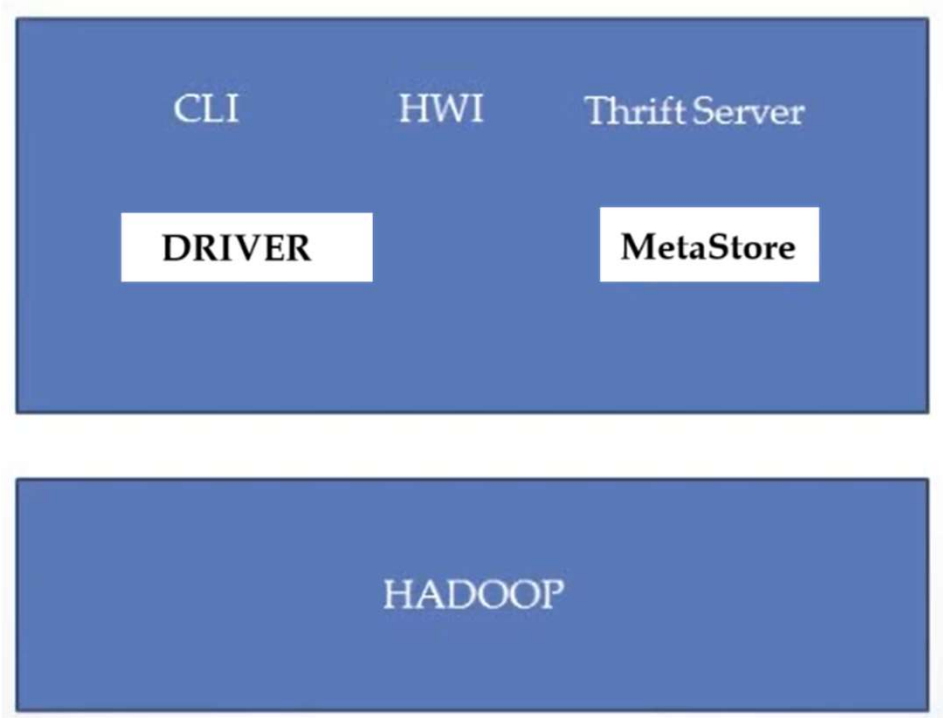


drwxr-xr-x	edureka	supergroup	0 B	12/14/2016, 3:29:26 PM	0	0 B	out_card
drwxr-xr-x	edureka	supergroup	0 B	12/14/2016, 3:34:22 PM	0	0 B	out_card1
drwxr-xr-x	edureka	supergroup	0 B	12/22/2016, 3:24:52 PM	0	0 B	output
-rw-r--r--	edureka	supergroup	136 B	12/23/2016, 1:16:35 PM	1	128 MB	piginput
drwxr-xr-x	edureka	supergroup	0 B	12/23/2016, 2:01:43 PM	0	0 B	pigresult
-rw-r--r--	edureka	supergroup	96.94 KB	12/20/2016, 9:16:07 PM	1	128 MB	sample.log
drwx------	edureka	supergroup	0 B	12/23/2016, 1:24:11 PM	0	0 B	tmp
drwxr-xr-x	edureka	supergroup	0 B	12/20/2016, 5:02:55 PM	0	0 B	user
drwxr-xr-x	edureka	supergroup	0 B	12/16/2016, 1:53:05 PM	0	0 B	weather_out

HIVE

- Hive Architecture
- Hive Components
- Metastore
- Limitations of Hive
- Where to use Hive
- Differences with traditional RDBMS
- Type System
- Hive Data Model

Architecture



Components

Shell - Environment

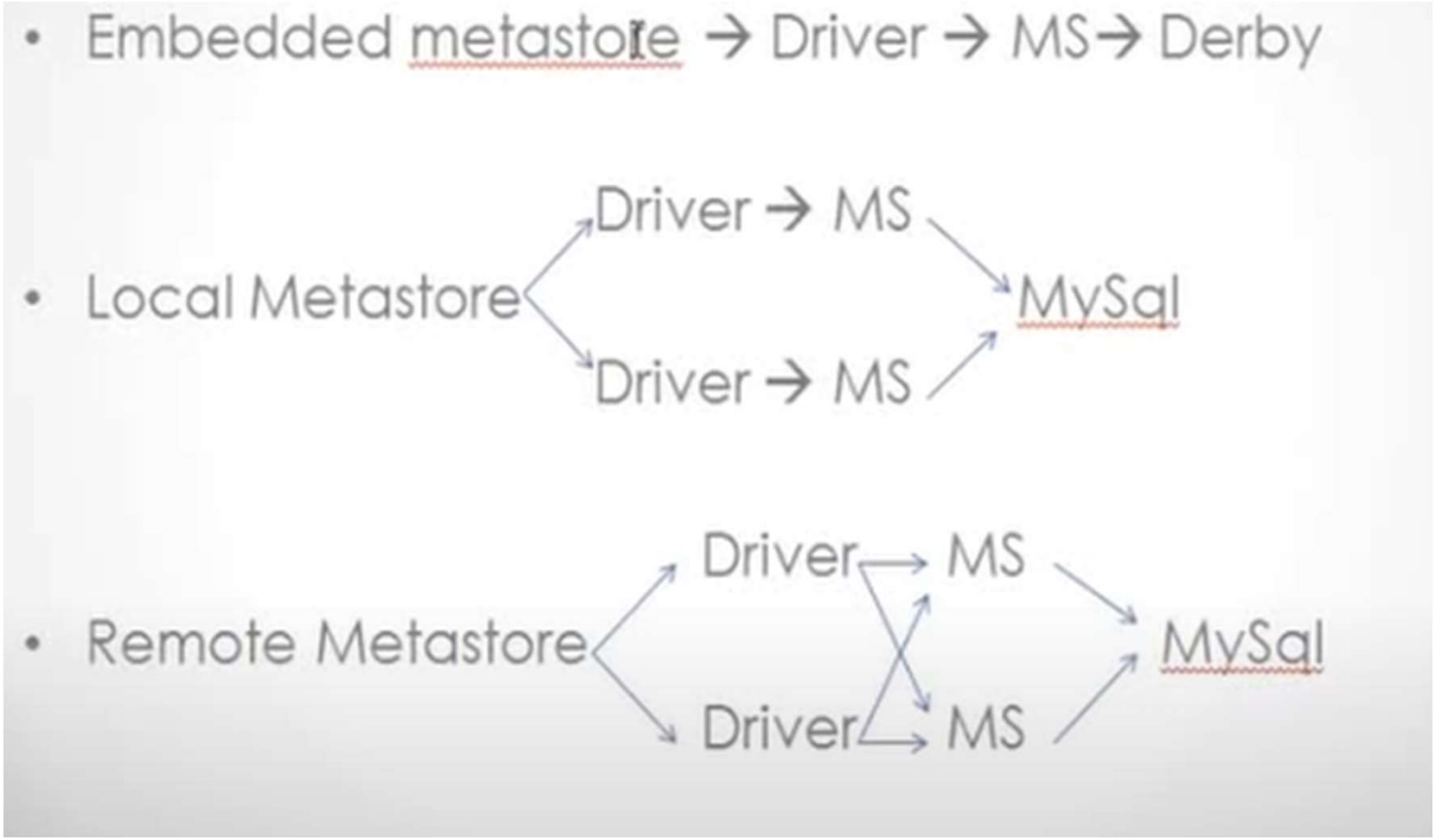
MetaStore - Metadata

Driver- Management of Life Cycle of Hql Statements

Compiler- Convert Hql to MR

Execution Engine- Execution of Job

Metastore


Embedded metastore → Driver → MS → Derby
Local Metastore
Driver → MS
Driver → MS
MySql
Remote Metastore
Driver
MS
Driver
MS
MySql

Limitations

- Not recommended for row level updates .
- Latency for Hive Queries is High
- Not designed for OLTP

Difference w.r.t RDBMS

Schema on Read V/s Schema on Write

Data Types

- Boolean :- true/false
- Integers :- TinyInt - 1 byte integer
  SmallInt - 2 byte integer
  INT - 4 byte integer
  BigInt - 8 byte integer
- Floating point :- Float / Double
- String

Data Types

NUMERIC TYPES	DESCRIPTION
TINYINT	1-byte signed integer, from `-128` to `127`
SMALLINT	2-byte signed integer, from `-32,768` to `32,767`
INT/INTEGER	4-byte signed integer, from -2,147,483,648 to 2,147,483,647
BIGINT	8-byte signed integer, from `-9,223,372,036,854,775,808` to `9,223,372,036,854,775,80`
FLOAT	4-byte single precision floating point number
DOUBLE	8-byte double precision floating point number
DOUBLE PRECISIO N	Alias for DOUBLE, only available starting with Hive 2.2.0
DECIMAL	It accepts a precision of 38 digits.
NUMERIC	Same as DECIMAL type.

DATE/TIME TYPES	DESCRIPTION
TIMESTAMP	Accepts Both Date and Time
DATE	Accepts just Date
INTERVAL	Interval

Data Types

STRING TYPES	DESCRIPTION
STRING	The string is an unbounded type. Not required to specify the lenght. It can accept max up to 32,767 bytes.
VARCHAR	Variable length of characters. It is bounded meaning you still need to specify the length like VARCHAR(10).
CHAR	Fixed length of Characters. if you define char(10) and assigning 5 chars, the remaining 5 characters space will be wasted.

Complex Data Types

- Struts
- Maps
- Array

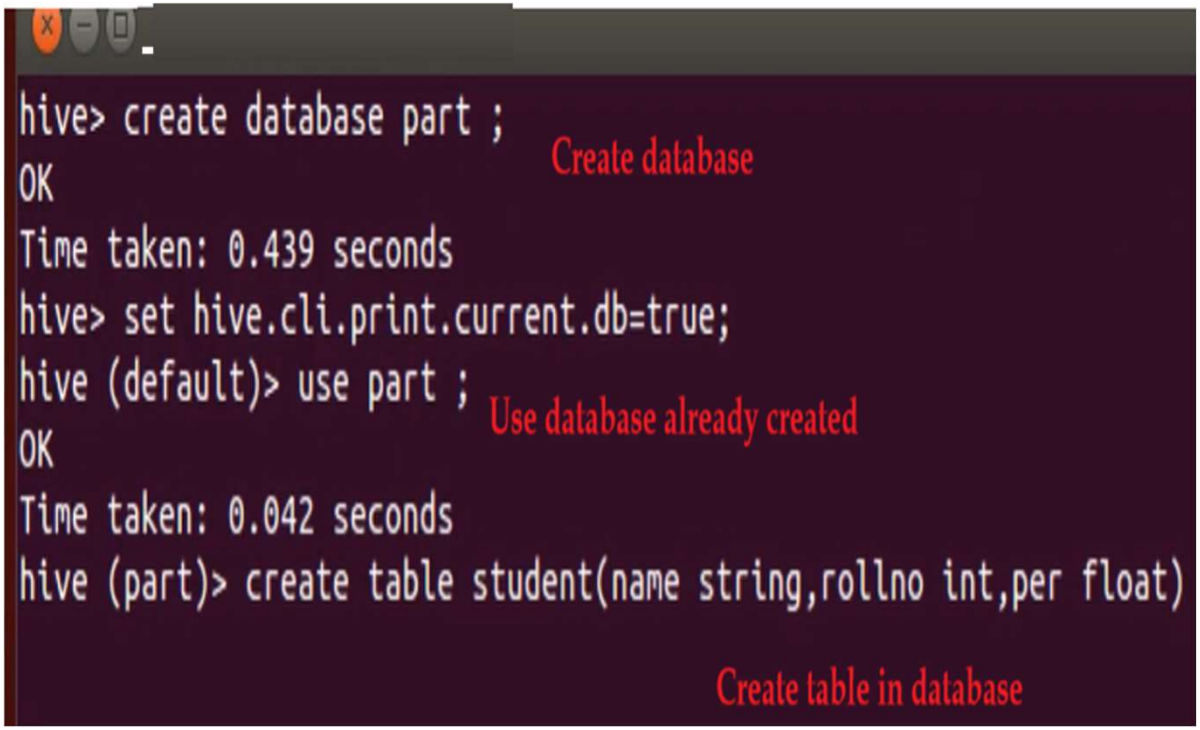
ACID -limitations

- To support ACID, Hive tables should be created with **TRANSACTIONAL** table property.
- Currently, Hive supports ACID transactions on tables that store **ORC** file format.
- Enable ACID support by setting transaction manager to **DbTxnManager**
- Transaction tables cannot be accessed from the non-ACID Transaction Manager (**DummyTxnManager**) session.
- **External tables** cannot be created to support ACID since the changes on external tables are beyond Hive control.
- **LOAD** is not supported on ACID transactional Tables. hence use **INSERT INTO**.
- On the Transactional session, all operations are auto commit as **BEGIN**, **COMMIT**, and **ROLLBACK** are not yet supported.

Data Models

- Database
- Table
- Partition
- Buckets & Clusters

Partitioning - Static



Partition Info

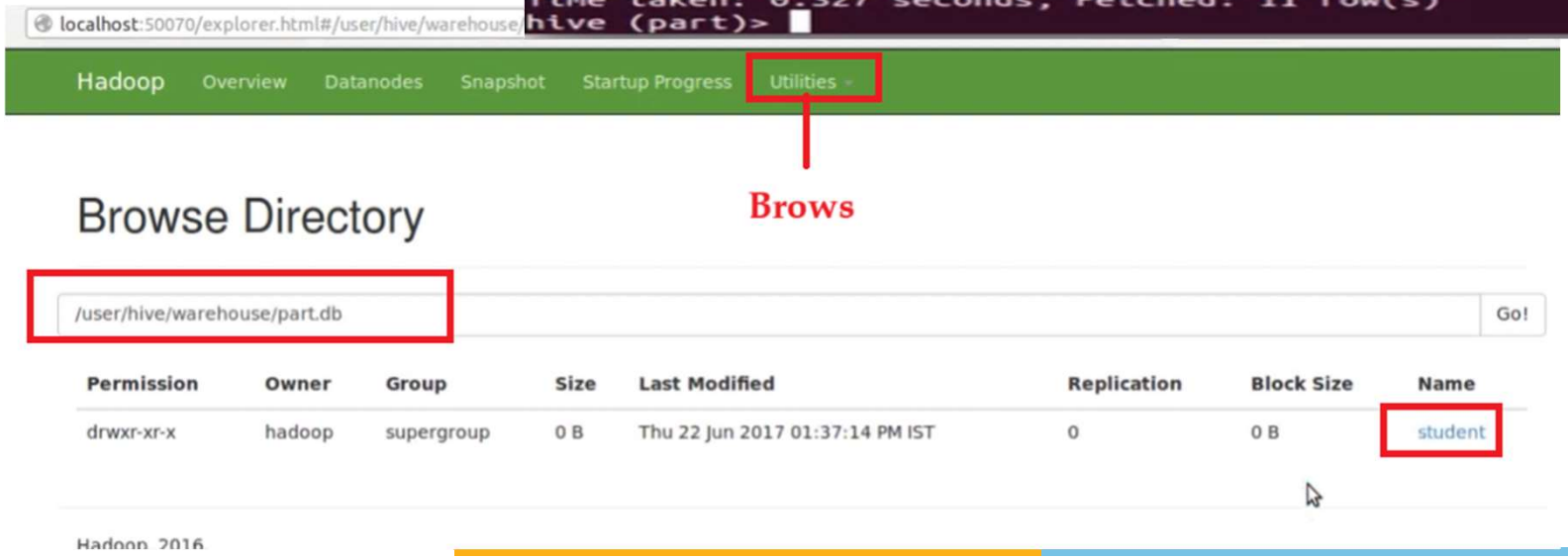
```
hive (default)> use part ;
OK
Time taken: 0.042 seconds
hive (part)> create table student(name string,rollno int,per float)
           > partitioned by (state string,city string)
           > row format delimited
           > fields terminated by ',';
OK
Time taken: 0.513 seconds
hive (part)> describe student ;
OK
name                    string
rollno                  int
per                     float
state                   string
city                    string

# Partition Information
# col_name              data_type               comment

state                   string
city                    string
Time taken: 0.327 seconds, Fetched: 11 row(s)
hive (part)>
```

Partition column

Partition Column



Inserting Data in Partitioned Table

Partitioning.txt × partstud mah ×

```
sdp,12,75.2
xcd,45,45.5
dss,47,74.2
fss75,90
qdz,5,32
dds,55,78.2
gff,66,85.4
jhh,22,65.02
sd,45,14.04
sf,15,78.87
vdf,22,65.21
fdf,24,75.25
sdsd,2,95.23
dsd,20,71.05
sfdf,29,69.00
```

Trying to insert this data from "partstud.mah"
(Name, Rollno, Percentage"

```
hive (part)> load data local inpath '/home/hadoop/Desktop/HadoopData/partstudmah' into table student
           > partition(state="maharashtra",city="aurangabad");
Loading data to table part.student partition (state=maharashtra, city=aurangabad)
OK
Time taken: 2.259 seconds
hive (part)>
```

Partition data

```
hive (part)> load data local inpath '/home/hadoop/Desktop/HadoopData/partstudmah' into table student
           > partition(state="maharashtra",city="aurangabad");
Loading data to table part.student partition (state=maharashtra, city=aurangabad)
OK
Time taken: 2.259 seconds
hive (part)> load data local inpath '/home/hadoop/Desktop/HadoopData/partstudgujrat' into table student
           > partition(state="gujrat",city="surat");
Loading data to table part.student partition (state=gujrat, city=surat)
OK
Time taken: 0.674 seconds
hive (part)>
```

second partition on partstudgujrat

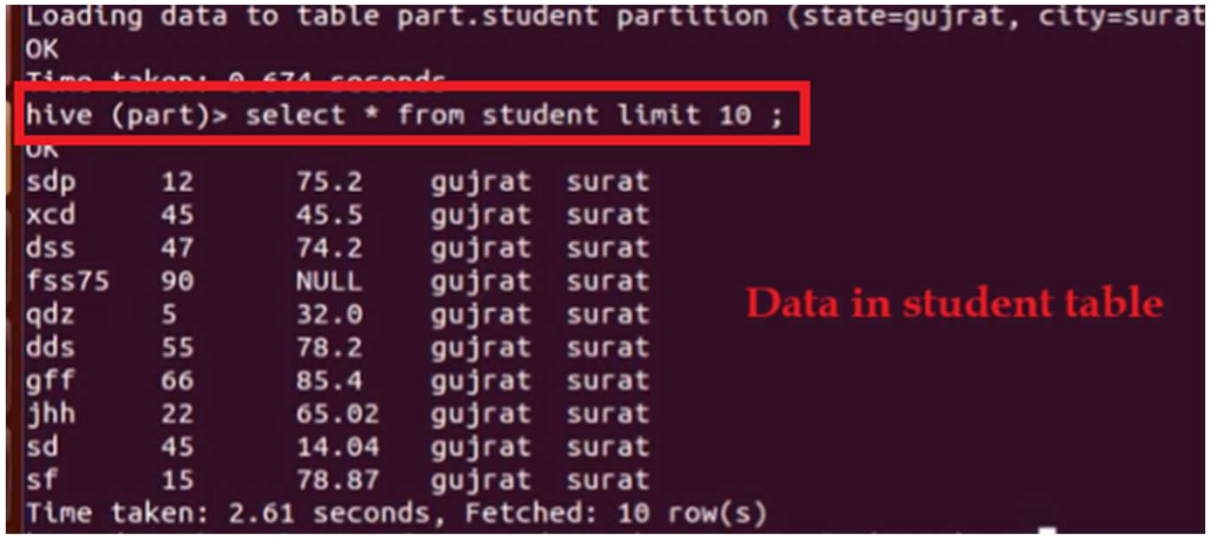
/user/hive/warehouse/part.db/student Go

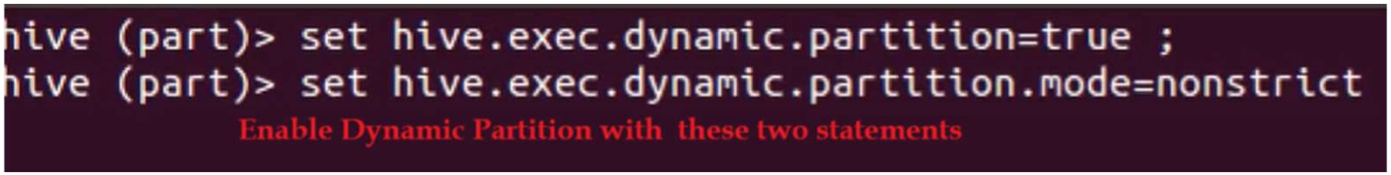
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Thu 22 Jun 2017 01:46:27 PM IST	0	0 B	state=gujrat
drwxr-xr-x	hadoop	supergroup	0 B	Thu 22 Jun 2017 01:43:47 PM IST	0	0 B	state=maharashtra

/user/hive/warehouse/part.db/student/state=gujrat/city=surat

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	hadoop	supergroup	2.93 KB	Thu 22 Jun 2017 01:46:27 PM IST	1	122.07 MB	partstudgujrat

Partitioning Dynamic





Load file data (3 files) –Static mode

```
hive (part)> set hive.exec.dynamic.partition=true ;
hive (part)> set hive.exec.dynamic.partition.mode=nonstrict;
hive (part)> create table student1(name string,rollno int,per float,state string,city string)
           > row format delimited
           > fields terminated by ',';
OK
Time taken: 0.13 seconds
hive (part)> load data local inpath '/home/hadoop/Desktop/HadoopData/partstud1' into table student
           > ;
FAILED: SemanticException [Error 10062]: Need to specify partition columns because the destination table is partitione
hive (part)> load data local inpath '/home/hadoop/Desktop/HadoopData/partstud1' into table student1;
Loading data to table part.student1
OK
Time taken: 0.399 seconds
hive (part)> load data local inpath '/home/hadoop/Desktop/HadoopData/partstud2' into table student1;
Loading data to table part.student1
OK
Time taken: 0.355 seconds
hive (part)> load data local inpath '/home/hadoop/Desktop/HadoopData/partstud3' into table student1;
```

loading data from student1 , student2 and student3

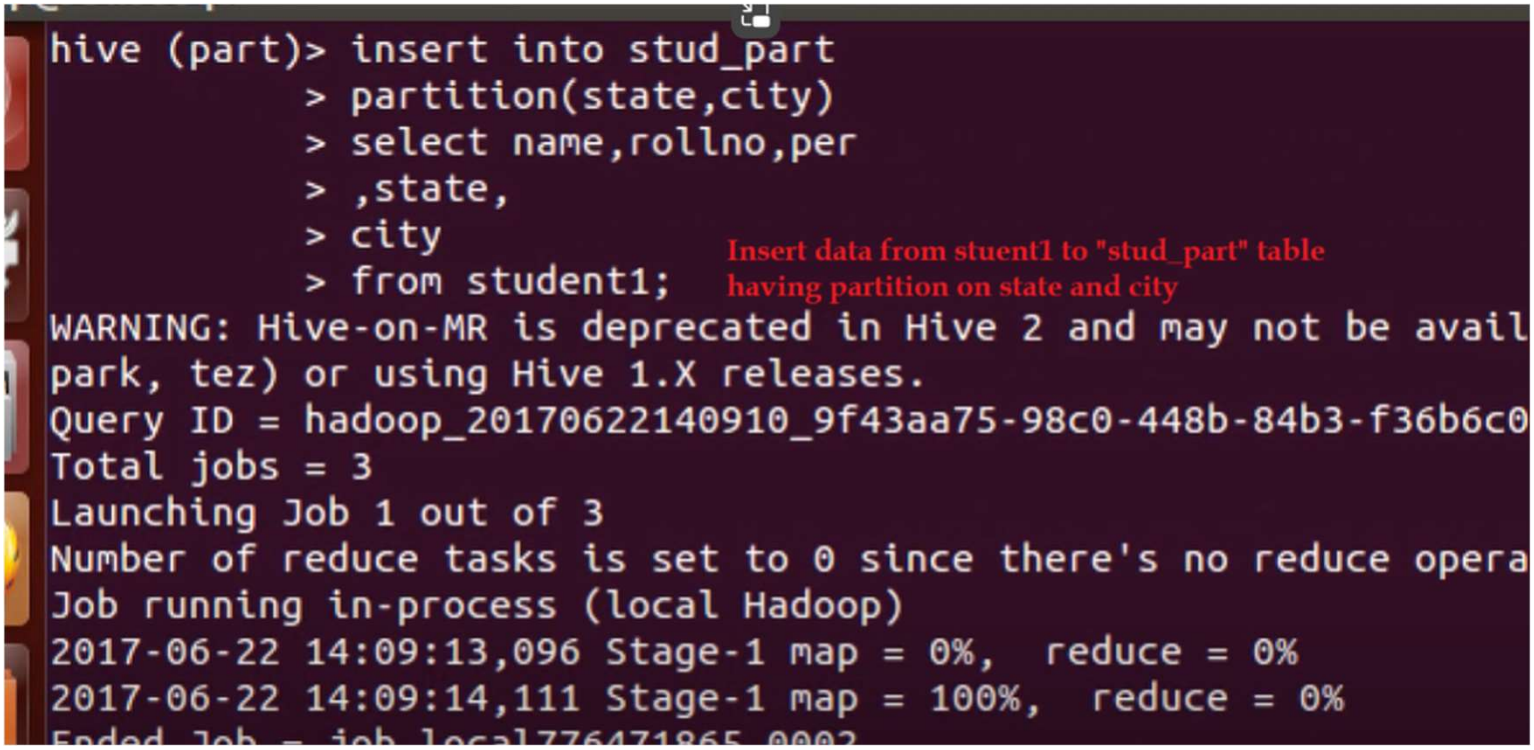
/user/hive/warehouse/part.db/student1 Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	hadoop	supergroup	8.84 KB	Thu 22 Jun 2017 01:59:44 PM IST	1	122.07 MB	partstud1
-rwxr-xr-x	hadoop	supergroup	8.84 KB	Thu 22 Jun 2017 01:59:53 PM IST	1	122.07 MB	partstud2
-rwxr-xr-x	hadoop	supergroup	8.84 KB	Thu 22 Jun 2017 02:00:05 PM IST	1	122.07 MB	partstud3

```
Time taken: 0.418 seconds
hive (part)> select count(*) from student1;
park, tez) or using Hive 1.X releases.
Query ID = hadoop_20170622140043_366dacff-7545-4d9d-98e6-5ace8bde5a8a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
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>
Job running in-process (local Hadoop)
2017-06-22 14:00:48,527 Stage-1 map = 0%,  reduce = 0%
2017-06-22 14:00:49,557 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1879476739_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 63236 HDFS Write: 63236 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
936
Time taken: 5.725 seconds, Fetched: 1 row(s)
```

Totally 936 records inserted

Insert data from table to partitioned table



Data Presentation in dynamic mode

/user/hive/warehouse/part.db Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Thu 22 Jun 2017 02:09:15 PM IST	0	0 B	stud_part
drwxr-xr-x	hadoop	supergroup	0 B	Thu 22 Jun 2017 01:46:27 PM IST	0	0 B	student

/user/hive/warehouse/part.db/stud_part Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Thu 22 Jun 2017 02:09:14 PM IST	0	0 B	state=gujrat
drwxr-xr-x	hadoop	supergroup	0 B	Thu 22 Jun 2017 02:09:14 PM IST	0	0 B	state=maharashtra

/user/hive/warehouse/part.db/stud_part/state=gujrat Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Thu 22 Jun 2017 02:09:13 PM IST	0	0 B	city=surat

/user/hive/warehouse/part.db/stud_part/state=gujrat/city=surat Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	hadoop	supergroup	5.94 KB	Thu 22 Jun 2017 02:09:13 PM IST	1	122.07 MB	000000_0

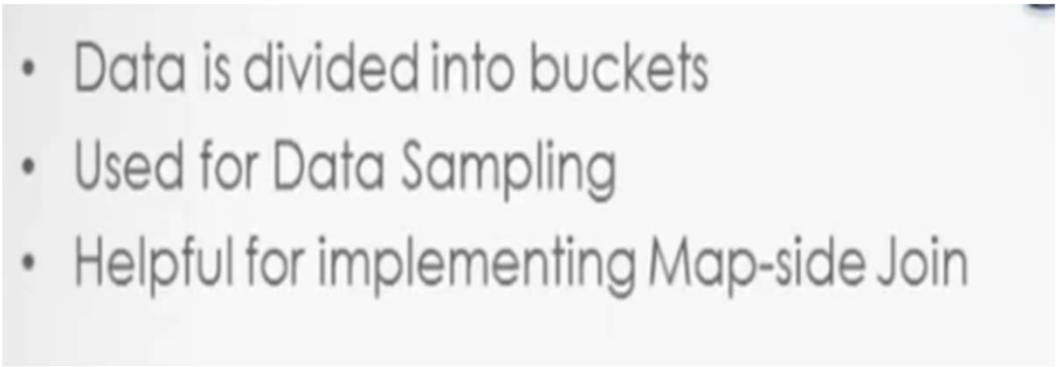
Transactional table


```
CREATE TABLE emp.employee_trans (
 id int,
 name string,
 age int,
 gender string)
 STORED AS ORC
 TBLPROPERTIES ('transactional'='true');
```

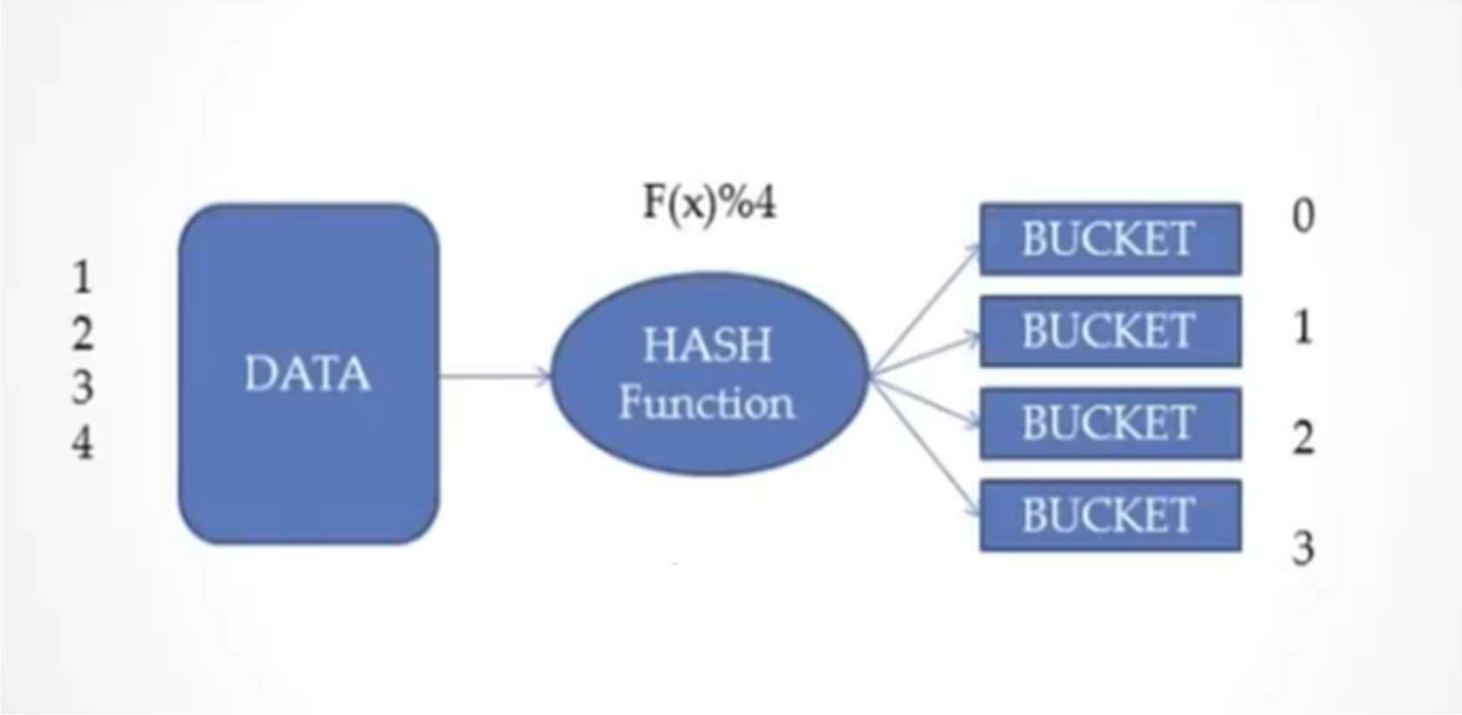
You can run the `DESCRIBE FORMATTED emp.employee` to check if the table is created with the transactional data_type as `TRUE`.

col_name	data_type	comment
# col_name	data_type	comment
id	int	
name	string	
age	int	
gender	string	
	NULL	NULL
# Detailed Table Information	NULL	NULL
Database:	emp	NULL
OwnerType:	USER	NULL
Owner:	prabha	NULL
CreateTime:	Sat Oct 17 04:00:05 UTC 2020	NULL
LastAccessTime:	UNKNOWN	NULL
Retention:	0	NULL
Location:	hdfs://192.168.1.148:9000/user/hive/warehouse/emp.db/employee_trans	NULL
Table Type:	MANAGED_TABLE	NULL
Table Parameters:	NULL	NULL
	bucketing_version	2
	numFiles	4
	totalSize	3356
	transactional	true
	transactional_properties	default
	transient_lastDdlTime	1602914461
	NULL	NULL
# Storage Information	NULL	NULL
SerDe Library:	org.apache.hadoop.hive.ql.io.orc.OrcSerde	NULL
InputFormat:	org.apache.hadoop.hive.ql.io.orc.OrcInputFormat	NULL
OutputFormat:	org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat	NULL

Bucketing


• Data is divided into buckets
• Used for Data Sampling
• Helpful for implementing Map-side Join

Hashing Technique



```
hadoop@Sandeep:~$ jps
4448 NodeManager
4135 SecondaryNameNode
4313 ResourceManager
3786 NameNode
3915 DataNode
8523 Jps
hadoop@Sandeep:~$ hive

Logging initialized using configuration in jar:file:/home/hadoop/Downloads/apache-hive-2.1.1-bin/lib/hiv
es Async: true
Java HotSpot(TM) Server VM warning: You have loaded library /home/hadoop/Downloads/hadoop-2.7.3/lib/nati
isabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noex
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a dif
) or using Hive 1.X releases.
hive> create database bucketeddb;
OK
Time taken: 1.597 seconds
hive> use bucketeddb;
OK
Time taken: 0.045 seconds
hive> create table sada_table1(id int,firstname string,lastname string)
    > row format delimited
    > fields terminated by ','
    > stored as textfile ;
OK
Time taken: 0.575 seconds
hive> load data local inpath '/home/hadoop/Desktop/HadoopData/bd1' into table sada_table ;
FAILED: SemanticException [Error 10001]: Line 1:72 Table not found 'sada_table'
hive> load data local inpath '/home/hadoop/Desktop/HadoopData/bd1' into table sada_table1 ;
Loading data to table bucketeddb.sada_table1
OK
Time taken: 1.163 seconds
hive>
```

Start Hadoop

Start Hive

Create database

Create simple table

load data in sada_table from bd1

```
hive> create table bucketwala_table1(id int,firstname string,lastname string)
    > clustered by (id) into 5 buckets
    > row format delimited
    > fields terminated by ','
    > stored as textfile;
OK
Time taken: 0.191 seconds
hive> insert overwrite table bucketwala_table1 select * from sada_table1 ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engi
park, tez) or using Hive 1.X releases.
Query ID = hadoop_20170823134810_55a07640-70f3-4d33-9e0a-afab8220cb96
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 5
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>
Job running in-process (local Hadoop)
2017-08-23 13:48:14,765 Stage-1 map = 100%,  reduce = 0%
2017-08-23 13:48:15,797 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1899406628_0001
Loading data to table bucketeddb.bucketwala_table1
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 2064 HDFS Write: 4374 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 6.241 seconds
```

Create bucketed table

insert data in bucketed table from sada_table

/user/hive/warehouse/bucketeddb.db

Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Wed 23 Aug 2017 01:48:16 PM IST	0	0 B	bucketwala_table1
drwxr-xr-x	hadoop	supergroup	0 B	Wed 23 Aug 2017 01:43:34 PM IST	0	0 B	sada_table1

/user/hive/warehouse/bucketeddb.db/sada_table1

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	hadoop	supergroup	344 B	Wed 23 Aug 2017 01:43:34 PM IST	1	122.07 MB	bd1

/user/hive/warehouse/bucketeddb.db/bucketwala_table1

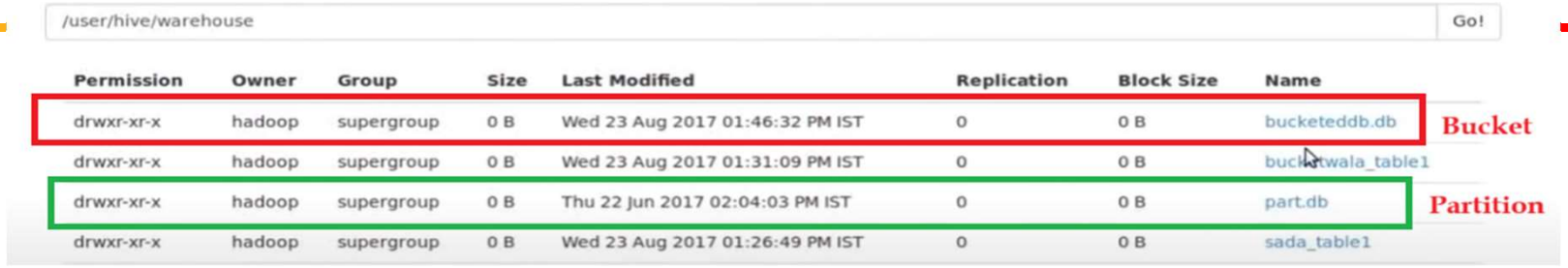
Go

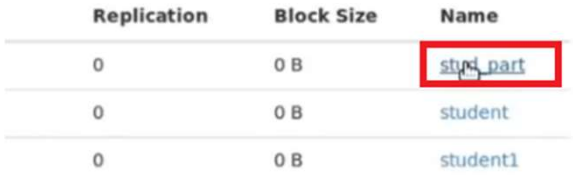
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	hadoop	supergroup	70 B	Wed 23 Aug 2017 01:48:14 PM IST	1	122.07 MB	000000_0
-rwxr-xr-x	hadoop	supergroup	75 B	Wed 23 Aug 2017 01:48:15 PM IST	1	122.07 MB	000001_0
-rwxr-xr-x	hadoop	supergroup	69 B	Wed 23 Aug 2017 01:48:15 PM IST	1	122.07 MB	000002_0
-rwxr-xr-x	hadoop	supergroup	63 B	Wed 23 Aug 2017 01:48:15 PM IST	1	122.07 MB	000003_0
-rwxr-xr-x	hadoop	supergroup	67 B	Wed 23 Aug 2017 01:48:15 PM IST	1	122.07 MB	000004_0

```
ble1/000000_0
Java HotSpot(TM) Server VM warning: You ha
ds/hadoop-2.7.3/lib/native/libhadoop.so.1
ard. The VM will try to fix the stack guar
It's highly recommended that you fix the
or link it with '-z noexecstack'.
17/08/23 13:51:20 WARN util.NativeCodeLoad
ry for your platform... using builtin-java
20,Sandeep,Patil
15,Ashwini,Vispute
10,Bhausaheb,Khamat
5,Vikas,Joshi
hadoop@Sandeep:~$
```

Data in Bucket ZERO

Partitioning V/s Bucketing





/user/hive/warehouse/bucketeddb.db Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Wed 23 Aug 2017 01:48:16 PM IST	0	0 B	bucketwala_table1
drwxr-xr-x	hadoop	supergroup	0 B	Wed 23 Aug 2017 01:43:34 PM IST	0	0 B	sada_table1

/user/hive/warehouse/bucketeddb.db/bucketwala_table1 Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	hadoop	supergroup	70 B	Wed 23 Aug 2017 01:48:14 PM IST	1	122.07 MB	000000_0
-rwxr-xr-x	hadoop	supergroup	75 B	Wed 23 Aug 2017 01:48:15 PM IST	1	122.07 MB	000001_0
-rwxr-xr-x	hadoop	supergroup	69 B	Wed 23 Aug 2017 01:48:15 PM IST	1	122.07 MB	000002_0
-rwxr-xr-x	hadoop	supergroup	63 B	Wed 23 Aug 2017 01:48:15 PM IST	1	122.07 MB	000003_0
-rwxr-xr-x	hadoop	supergroup	67 B	Wed 23 Aug 2017 01:48:15 PM IST	1	122.07 MB	000004_0

Table Types

- Internal Table
- External Table

Internal/Managed & External Table

Internal tables are also known as Managed tables that are owned and managed by Hive. By default, Hive creates a table as an Internal table and owned the table structure and the files.

In other words, Hive completely manages the lifecycle of the table (metadata & data) similar to tables in RDBMS.

For Internal tables, Hive by default stores the files at the data warehouse location which is located at `/user/hive/warehouse`

When you drop an internal table, it drops the data and also drops the metadata of the table.

Below is an example of creating internal table.

Internal / Managed Table


```
CREATE TABLE IF NOT EXISTS emp.employee (
 id int,
 name string,
 age int,
 gender string )
 COMMENT 'Employee Table'
 ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ',';
```

Use `DESCRIBE FORMATTED emp.employee;` to get the description of the table and you should see Table Type as `MANAGED TABLE`.

```
+-------------------------------+----------------------------------------------------+-----------------------------+
|           col_name            |                     data_type                      |           comment           |
+-------------------------------+----------------------------------------------------+-----------------------------+
| # col_name                    | data_type                                          | comment                     |
| id                            | int                                                |                             |
| name                          | string                                             |                             |
| age                           | int                                                |                             |
| gender                        | string                                             |                             |
|                               | NULL                                               | NULL                        |
| # Detailed Table Information  | NULL                                               | NULL                        |
| Database:                     | emp                                                | NULL                        |
| OwnerType:                    | USER                                               | NULL                        |
| Owner:                        | prabha                                             | NULL                        |
| CreateTime:                   | Thu Oct 15 04:58:36 UTC 2020                       | NULL                        |
| LastAccessTime:               | UNKNOWN                                            | NULL                        |
| Retention:                    | 0                                                  | NULL                        |
| Location:                     | hdfs://192.168.1.148:9000/user/hive/warehouse/emp.db/employee | NULL             |
| Table Type:                   | MANAGED_TABLE                                      | NULL                        |
| Table Parameters:             | NULL                                               | NULL                        |
```

External Table

Data in External tables are not owned or managed by Hive. To create an External table you need to use `EXTERNAL` clause.

Hive default stores external table files also at Hive managed data warehouse location but recommends to use external location using `LOCATION` clause.

Dropping an external table just drops the metadata but not the actual data. The actual data is still accessible outside of Hive.

Below is an example of creating an external table in Hive. If you noticed we use `EXTERNAL` and `LOCATION` options.

External Table


```
CREATE EXTERNAL TABLE emp.employee_external (
 id int,
 name string,
 age int,
 gender string)
 ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ','
 LOCATION '/user/hive/data/employee_external';
```

Use `DESCRIBE FORMATTED emp.employee_external;` to get the description of the table and you should see Table Type as `EXTERNAL TABLE`.

col_name	data_type	comment
# col_name	data_type	comment
id	int	
name	string	
age	int	
gender	string	
	NULL	NULL
# Detailed Table Information	NULL	NULL
Database:	emp	NULL
OwnerType:	USER	NULL
Owner:	prabha	NULL
CreateTime:	Thu Oct 15 05:08:10 UTC 2020	NULL
LastAccessTime:	UNKNOWN	NULL
Retention:	0	NULL
Location:	hdfs://192.168.1.148:9000/user/hive/warehouse/emp.db/employee_external	NULL
Table Type:	EXTERNAL_TABLE	NULL
Table Parameters:	NULL	NULL
	COLUMN_STATS_ACCURATE	{\"BASIC_STATS\":\"true\",\"CC

Dropping table

Regardless of the Internal and external table, Hive manages the table definition and its partition information in Hive Metastore. Dropping an internal table deletes the table metadata from Metastore and also removes all its data/files from HDFS.

Dropping an external table, just drop the metadata of the table from Metastore and keeps the actual data as-is om HDFS location.

Key differences -- Summary

INTERNAL OR MANAGED TABLE	EXTERNAL TABLE
By default, Hive creates an Internal or Managed Table.	Use EXTERNAL option/clause to create an external table
Hive owns the metadata, table data by managing the lifecycle of the table	Hive manages the table metadata but not the underlying file.
Dropping an Internal table drops metadata from Hive Metastore and files from HDFS	Dropping an external table drops just metadata from Metastore with out touching actual file on HDFS.
Hive supports ARCHIVE, UNARCHIVE, TRUNCATE, MERGE, CONCATENATE operations	Not supported
Supports ACID/Transactional	Not supported
Supports result caching	Not supported

Hadoop…Hbase….Hive….Pig

Hadoop:

- Hadoop is basically 2 things: a **Distributed FileSystem (HDFS) + Computation or Processing framework (MapReduce).**
- Like all other FS, HDFS also provides us storage, but in a fault tolerant manner with high throughput and lower risk of data loss (because of the replication). But, being a FS, HDFS lacks **random read and write access**.
- This is where HBase comes into picture. It's a **distributed, scalable, big data store**, modelled after Google's BigTable. It stores data as key/value pairs.

HBase:

- Apache HBase is an open source NoSQL database that provides real-time read/write access to those large datasets.
- HBase scales linearly to handle huge data sets with billions of rows and millions of columns, and it easily combines data sources that use a wide variety of different structures and schemas.
- HBase is natively integrated with Hadoop and works seamlessly alongside other data access engines through YARN.

Hive:

- It provides us data warehousing facilities on top of an existing Hadoop cluster. Along with that it provides an SQL like interface which makes work easier.
- While Pig is basically a **dataflow language** that allows us to process enormous amounts of data very easily and quickly.
- Pig basically has 2 parts: the Pig **Interpreter** and the language, **PigLatin**. We can write Pig script in PigLatin and using Pig interpreter process them.
- Pig makes our life a lot easier, otherwise writing MapReduce is always not easy.

Pig V/s Hive

S.No.	Pig	Hive
1.	Pig operates on the client side of a cluster.	Hive operates on the server side of a cluster.
2.	Pig uses pig-latin language.	Hive uses HiveQL language.
3.	Pig is a Procedural Data Flow Language.	Hive is a Declarative SQLish Language.
4.	It was developed by Yahoo.	It was developed by Facebook.
5.	It is used by Researchers and Programmers.	It is mainly used by Data Analysts.
6.	It is used to handle structured and semi-structured data.	It is mainly used to handle structured data.
7.	It is used for programming.	It is used for creating reports.
8.	Pig scripts end with .pig extension.	In HIve, all extensions are supported.
9.	It does not support partitioning.	It supports partitioning.
10.	It loads data quickly.	It loads data slowly.
11.	It does not support JDBC.	It supports JDBC.
12.	It does not support ODBC.	It supports ODBC.
13.	Pig does not have a dedicated metadata database.	Hive makes use of the exact variation of dedicated SQL-DDL language by defining tables beforehand.

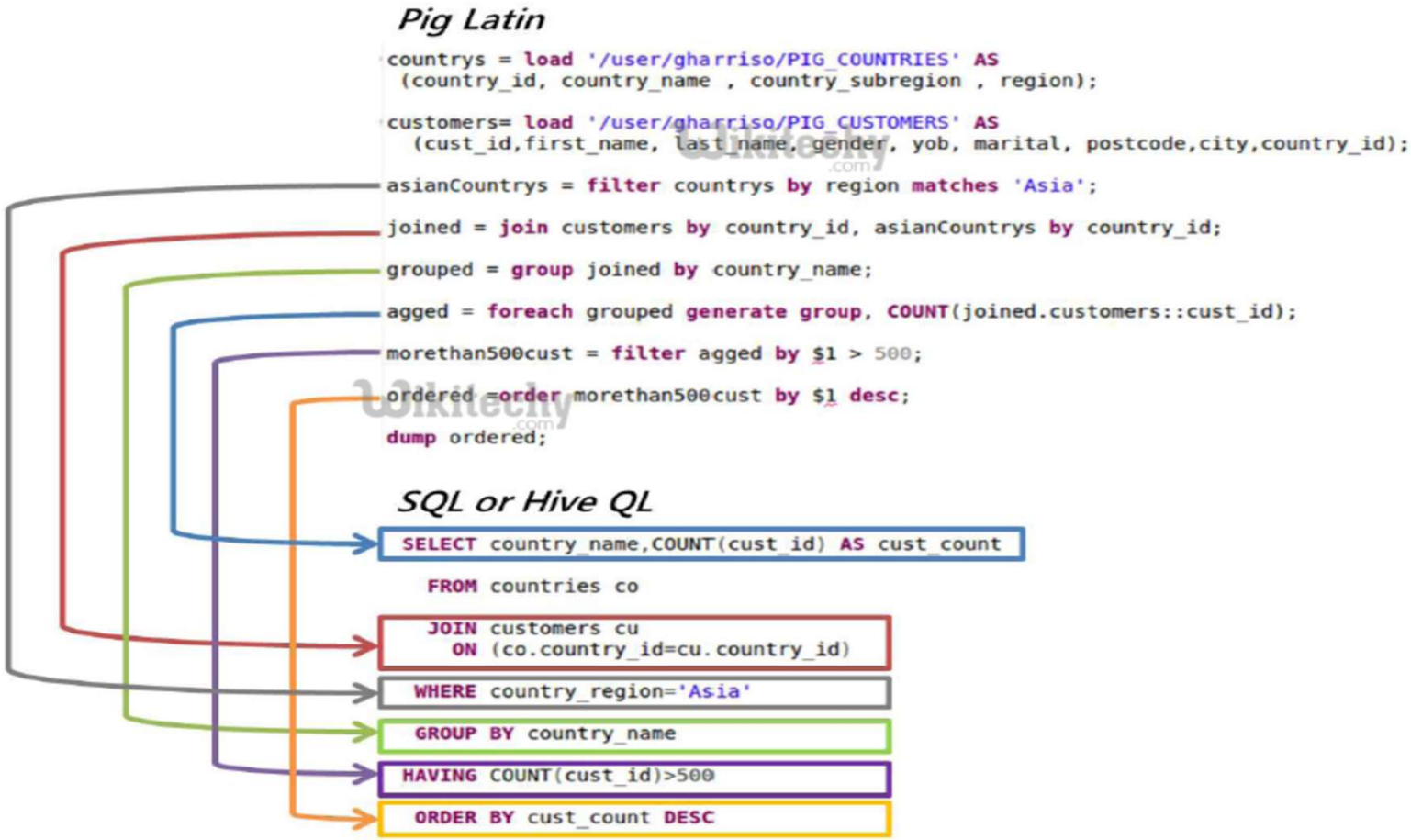
Hive V/s HBase

S. No.	Parameters	Hive	HBase
1.	Basics	Hive is a query engine that uses queries that are mostly similar to SQL queries.	It is Data storage, particularly for unstructured data.
2.	Used for	It is mainly used for batch processing (that means OLAP-based).	It is extensively used for transactional processing (that means OLTP).
3.	Processing	It cannot be used for real-time processing since immediate analysis results are unable to obtain. In other words, the operations in Hive require batch processing, they normally take a long time to complete.	It can be used to process data in real-time. Transactional operations are faster than non-transactional operations (since HBase stores data in the form of key-value pairs).
4.	Queries	It is used only for analytical queries. It is mostly used to analyze Big Data.	It is used for real-time querying. It is mostly used to query Big Data.
5.	Runs on	Hive runs on the top of Hadoop.	HBase runs on the top of HDFS (Hadoop Distributed File System).
6.	Database	Apache Hive is not a database.	It supports the NoSQL database.
7.	Schema	It has a schema model.	It is free from the schema model.

Contd…

8.	Latency	Made for high latency operations as batch processing takes time.	Made for low-level latency operations.
9.	Cost	It is expensive as compared to HBase.	It is cost-effective as compared to Hive.
10.	Query Language	Hive uses HQL (Hive Query Language).	To conduct CRUD (Create, Read, Update, and Delete) activities, HBase does not have a specialized query language. HBase includes a Ruby-based shell where you can use Get, Put, and Scan functions to edit your data.
11.	Level of Consistency	Eventual consistency	Immediate consistency
12.	Secondary Indexes	It does not support Secondary Indexes.	It supports Secondary Indexes.
13.	Example	Hubspot	Facebook

Clear Difference



THAT IS ALL