LEARN.
CONNECT.
ELEVATE.

YCBS 257 - Data at Scale (Winter 2019)

Instructor: Khaled Tannir

McGill | School of Continuing Studies

mcgill.ca
/continuingstudies

School of Continuing Studies

YCBS 257 - Data at Scale (BIG DATA)

Course 5

Hadoop Analyzing Tools

Khaled Tannir

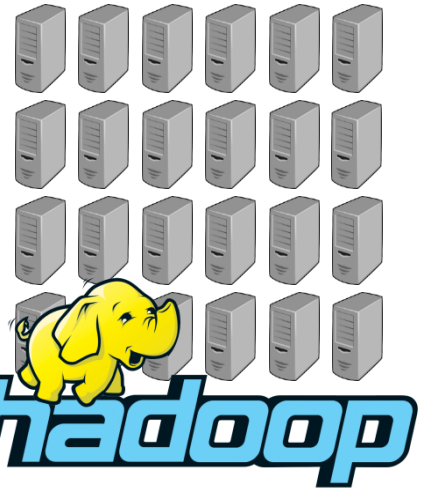
McGill

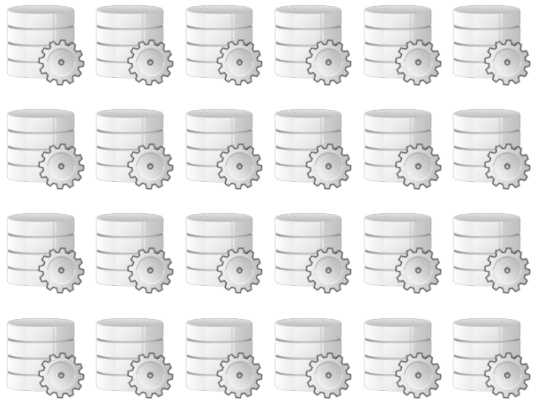
Theme of this Course

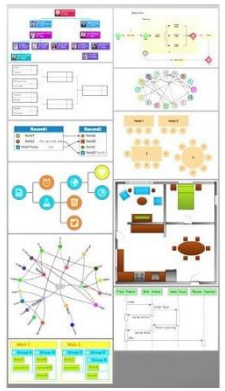
Data Analyzing Tools

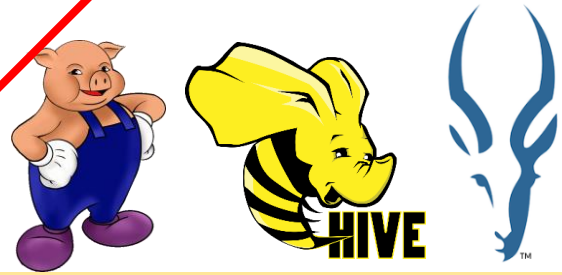
- *Apache HIVE*
  - *Core Concepts, HiveQL query Language*
- *Apache Impala*
  - *Core Concepts*
- *Apache HCatalog*
  - *Core Concepts, Sharing metadata between Hive, Pig …*

Machine Learning at Scale



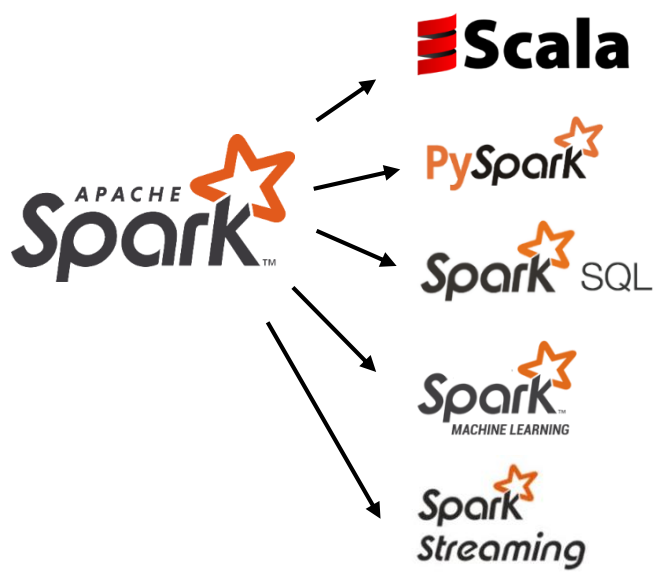






Analyze

Ingest



Apache HIVE

Data Warehousing & Analytics on Hadoop

McGill

FALL 2018

What is Apache Hive?

- A database/data warehouse on top of Hadoop
- Rich data types (structs, lists and maps)
- Can deal with different storage and file formats.
- Provides HQL (SQL like Query Language)

Efficient implementations of SQL filters, joins and group-by's on top of map reduce

- Summarize Big Data, and makes querying and analyzing easy.
- Written in Java and open-source

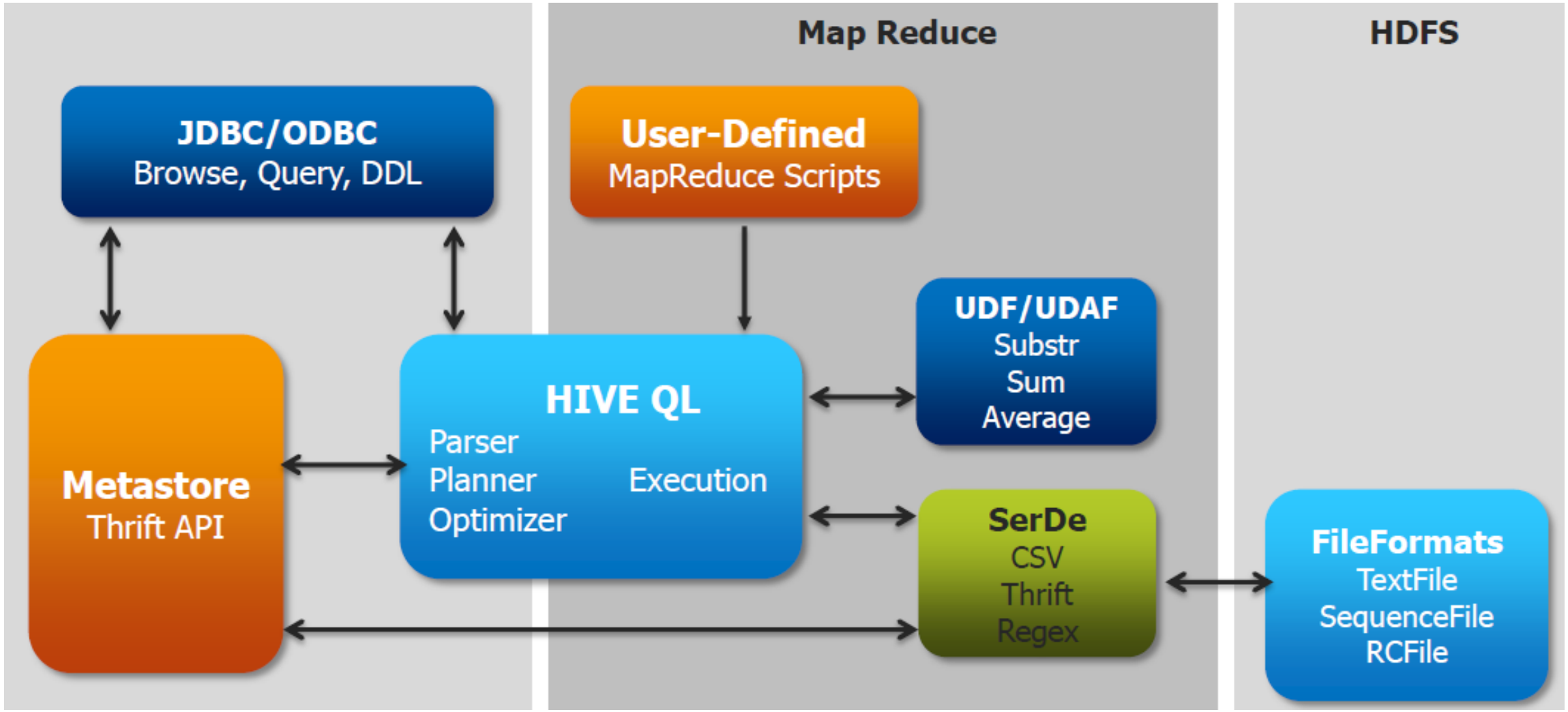
Apache Hive Origin

- Hive was Initially developed by Facebook where a huge volume of data need to be processed everyday.
- Looking for alternatives to MapReduce and Pig
- Map-reduce hard to program (users know sql/bash/python)
- Need to publish data in well known schemas

How it Works?

- **Hive is built on top of Hadoop**
  - *Uses Map-Reduce for execution*
  - *HDFS for storage – but any system that implements Hadoop FS API*
- **Hive compile SQL Quires into Map Reduce jobs and run the jobs in the Hadoop cluster**

Hive High Level Architeture



Hive Internal Components

- **Compiler and Planner**

  It compiles and checks the input query and create an execution plan.

- **Optimizer**

  It optimizes the execution plan before it runs

- **Execution Engine**

  Runs the Execution plan . It is guaranteed that execution plan is DAG.

Hive Metastore

- **Database**

  Namespace containing a set of tables

- **Table**

  Contains list of columns and their types and serDe infos

- **Partition**

  Each partition can have its own columns , SerDe and storage info
  Mapping to HDF Directories

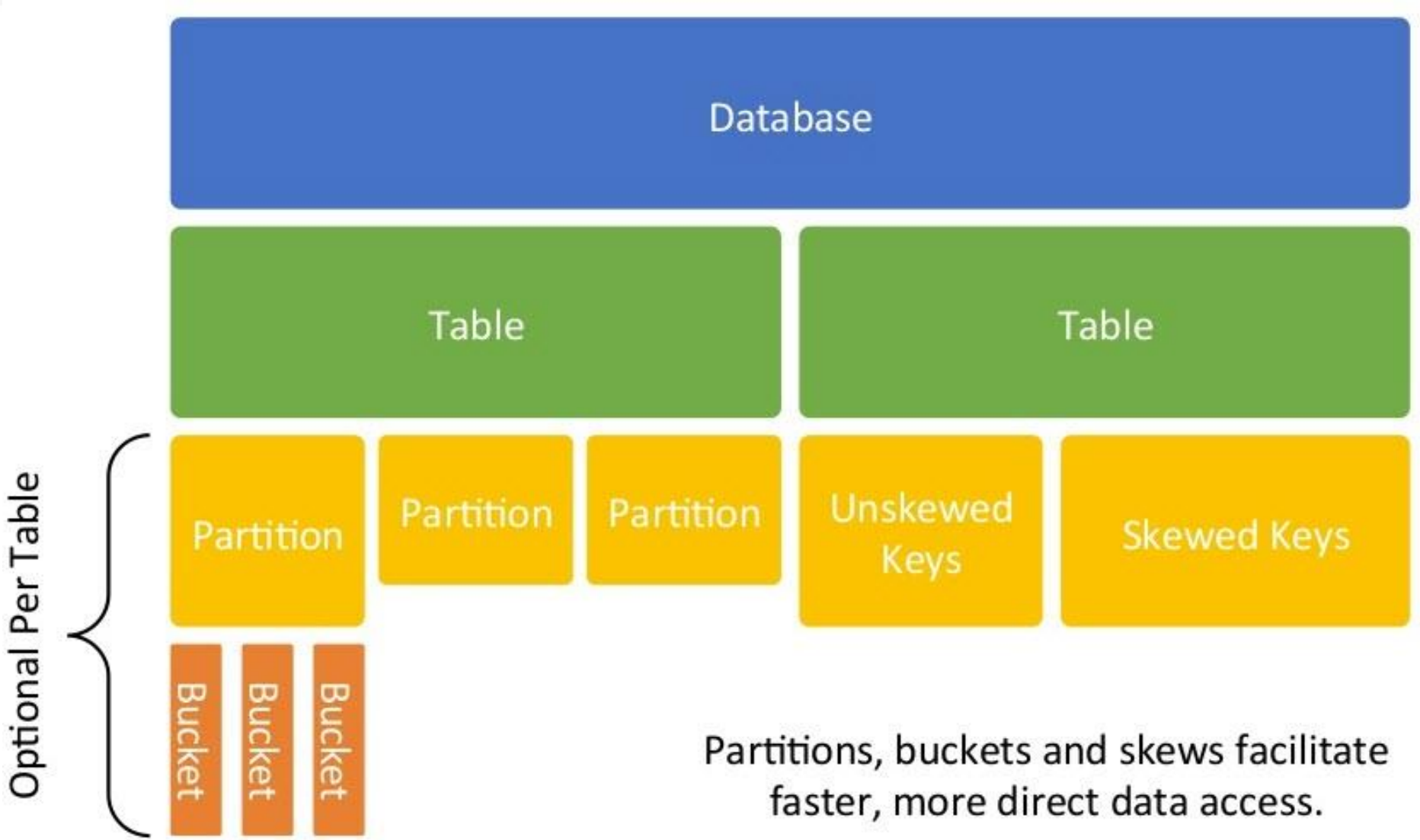
- **Statistics**

  Info about the databse

Hive Data Model

- **Tables**
  - *Analogous to relational tables*
  - *Each table has a corresponding directory in HDFS*
  - *Data serialized and stored as files within that directory*
- **Partitions**
  - *Each table can be broken into partitions*
  - *Partitions determine distribution of data within subdirectories*
- **Buckets**
  - *Data in each partition divided into buckets*
  - *Based on a hash function of the column*
  - *Each bucket is stored as a file in partition directory*

Data Abstraction in HIve



HIVE Internal (Managed) Tables

- **First we have to create table and load the data**
  - **Data on Schema**

- **Both data and schema will be removed if the table is dropped.**

- **We use Internal table :**
  - *When data is temporary*
  - *Data is not needed after deletion*
  - *If Hive is using the table data completely (no any external sources like pig, sqoop, mapreduce, …use the table).*

HIVE External (Non-Managed) Tables

- **The data is available on HDFS and table is created on HDFS data**
  - **Schema on Data.**

- **Only schema will be dropped the time of dropping the table, as data will be still available in the HDFS as before.**
- **We use external tables:**
  - *When data is available in HDFS*
  - *When files are being used outside of Hive*

Hive Persistence Formats

- **Built-in Formats:**
  - *ORC File, RCFile* (Optimized Row Columnar, Row Columnar)
  - *Avro*
  - *Delimited Text*
  - *Sequence File*
  - …
- **3rd-Party Addons:**
  - *JSON*
  - *XML*

Loading Data in Hive

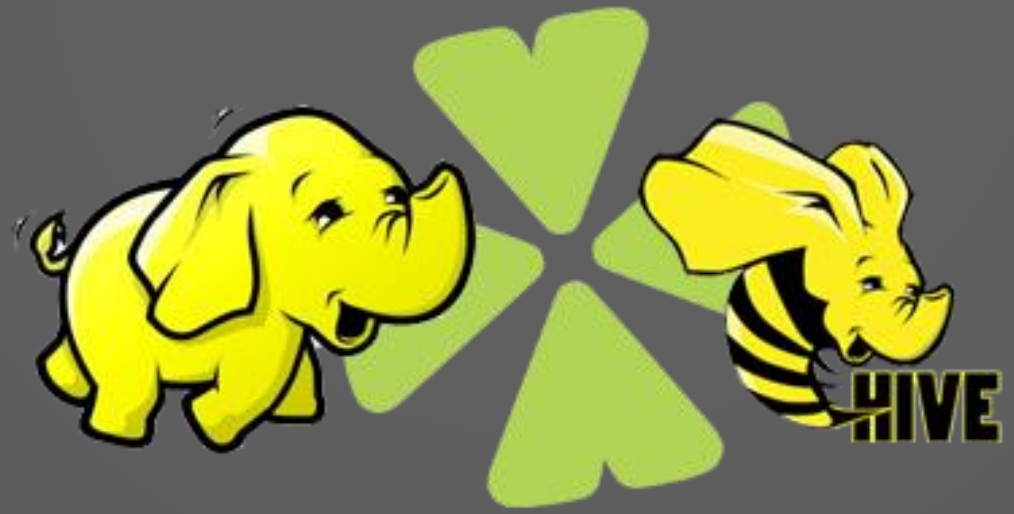
- **Hive LOAD**
  - *Load files from HDFS or local file system.*
  - *Format must agree with table format.*
- **Insert from query**
  - *CREATE TABLE AS SELECT or INSERT INTO.*
- **WebHDFS + WebHCat**
  - *Load data via REST APIs.*
- **Sqoop**
  - *Data transfer from external RDBMS to Hive.*
  - *Sqoop can load data directly to/from HCatalog.*

Differentiating HIVE from PIG

HIVE	PIG
Best for structured Data	Best for semi structured data
Its used for reporting	Used for programming
Hive supports partitions	Pig doesn’t supports partitions
It can start an optional thrift based server	It cannot start an optional thrift based server
It defines tables beforehand (schema) and stores schema information in a database	PIG doesn't have a dedicated metadata of database
Language: Hive is Declarative	Language: PIG is a procedural data-flow language.
Debugging HIVE code in local is complex and time consuming.	PIG code can be debugged in Local

HiveQL

Overview of the SQL on Hadoop language



Hive Query Language

- Select
- FROM
- WHERE
- GROUP BY
- HAVING
- JOIN

Complex Types: Complex types can be constructed using primitive data types :

- **Structs**
- **Maps** or key value pairs
- **Arrays** – Indexed lists

Primitive types:

- INTEGERS
  - TINY INT 1 byte integer
  - SMALL INT 2 byte integer
  - INT 4 byte integer
  - BIGINT 8 byte integer
- Date Type
- BOOLEAN
  - BOOLEAN TRUE or FALSE
- FLOATING POINT numbers
  - FLOAT Single precision
  - DOUBLE Double precision
- STRING type
  - STRING Sequence of characters

HiveQL : Examples

- hive> show tables;
- hive> create table SHAKESPEARE (freq INT,word STRING) row format delimited fields terminated by '\t' stored as textfile
- hive> load data inpath "shakespeare_freq" into table shakespeare;
- hive> select * from shakespeare where freq >100 sort by freq asc limit 10;

HiveQL : Create Table Examples

```
create table product_dtl (
        product_id: int,
        product-name: string,
        product_price: float,
        product_category: string)
rows format
delimited
fields terminated by ',' ;
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS Cars (
      Name STRING,  Miles_per_Gallon INT,
       Cylinders INT,   Displacement INT,
       Horsepower INT,  Weight_in_lbs INT,
       Acceleration DECIMAL,   Year DATE,
       Origin CHAR(1))
   COMMENT 'Data about cars from a public database'
   ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ','
   STORED AS TEXTFILE
Location '/user/<username>/visdata';
```

HiveQL : Loading Data


```
LOAD DATA [LOCAL] INPATH /tmp/pv_2008-06-08_us.txt INTO
TABLE page_view [PARTITION(date='2008-06-08', country='US')];
```

```
INSERT OVERWRITE [LOCAL] DIRECTORY '/tmp/pv_gender_sum'
SELECT pv_gender_sum.* FROM pv_gender_sum;
```

HiveQL : Sampling

```
INSERT OVERWRITE TABLE pv_gender_sum_sample
SELECT pv_gender_sum.*
FROM pv_gender_sum TABLESAMPLE (BUCKET 3 OUT OF 32);
```

Running Hive

- To run hive, enter this command in a Terminal window

```
$ hive
```

Or

(Recommended)

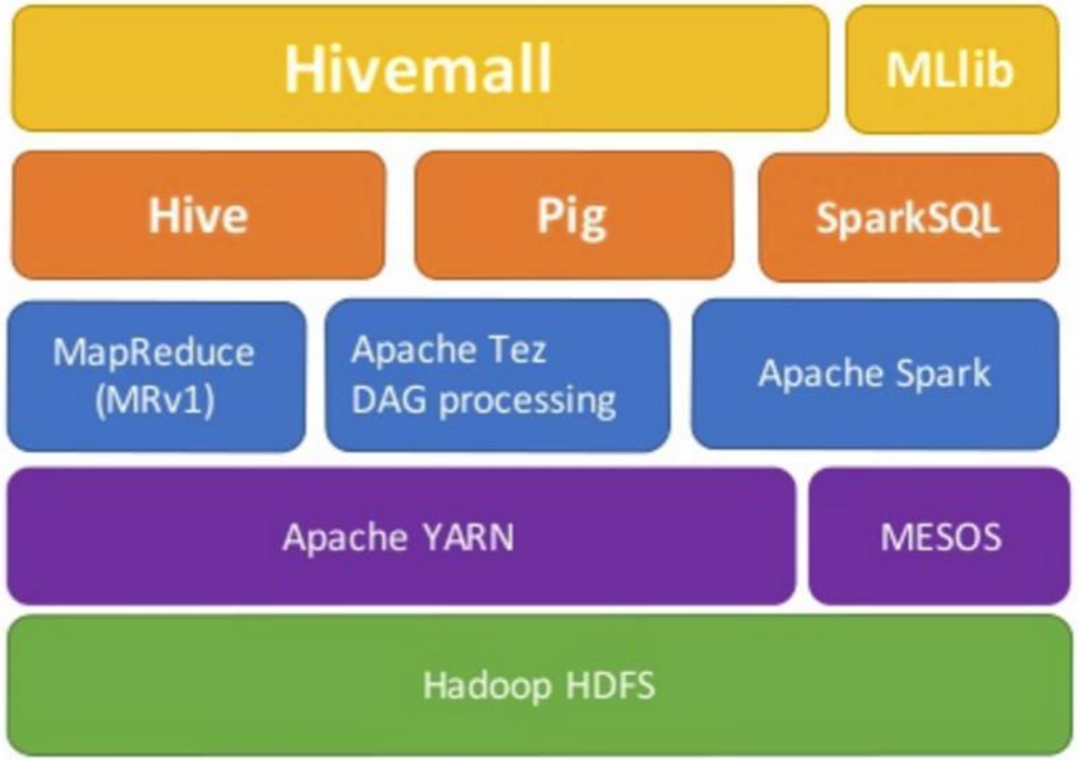
```
$ beeline -u jdbc:hive2://localhost:10000
```

- To run a Hive script

```
$ hive  -f  /home/cloudera/mysql.hql
```

Apache HiveMall

- A scalable machine learning library built as a collections of Hive UDFs.
- Supported Algorithms:
  - Binary Classification
  - Multi-class Classification
  - Regression
  - Recommendation
  - k-Nearest Neighbor
  - Anomaly Detection
  - Natural Language Processing (English/Japanese)



https://hivemall.incubator.apache.org/

Apache Impala

Diving into the Hadoop core

McGill

FALL 2018

What is Apache Impala?

- General-purpose SQL engine
- Real-Time queries in Apache Hadoop
- Runs directly within Hadoop
- Released in 2010 by Cloudera
- Based on Google's Dremel paper
- Open source under Apache license

Impala: Goals

- **General-purpose SQL query engine for Hadoop**
- **High performance**
  - *C++ implementation*
  - *runtime code generation (using LLVM*)*
  - *direct data access (no MapReduce jobs)*
- **Run directly on Hadoop**
  - *read the same file formats*
  - *use the same storage managers (Hive metastore)*
  - *daemons on the same nodes that run Hadoop processes*

* https://en.wikipedia.org/wiki/LLVM

Impala Data formats

- Supported HDFS file formats
  - Parquet
  - Text
  - Avro*
  - RCFile*
  - SequenceFile*

* no inserts, use Hive for that

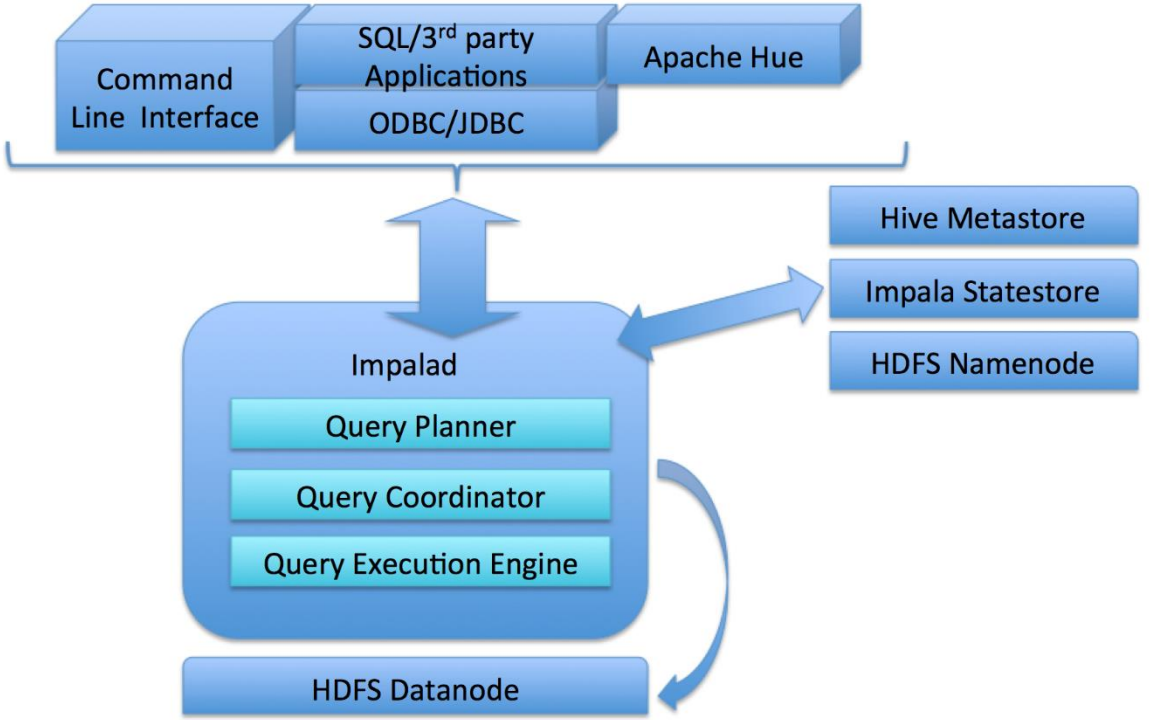
- Querying HBase tables and Amazon S3 Filesystem is also possible

Impala metadata and Hive metastore

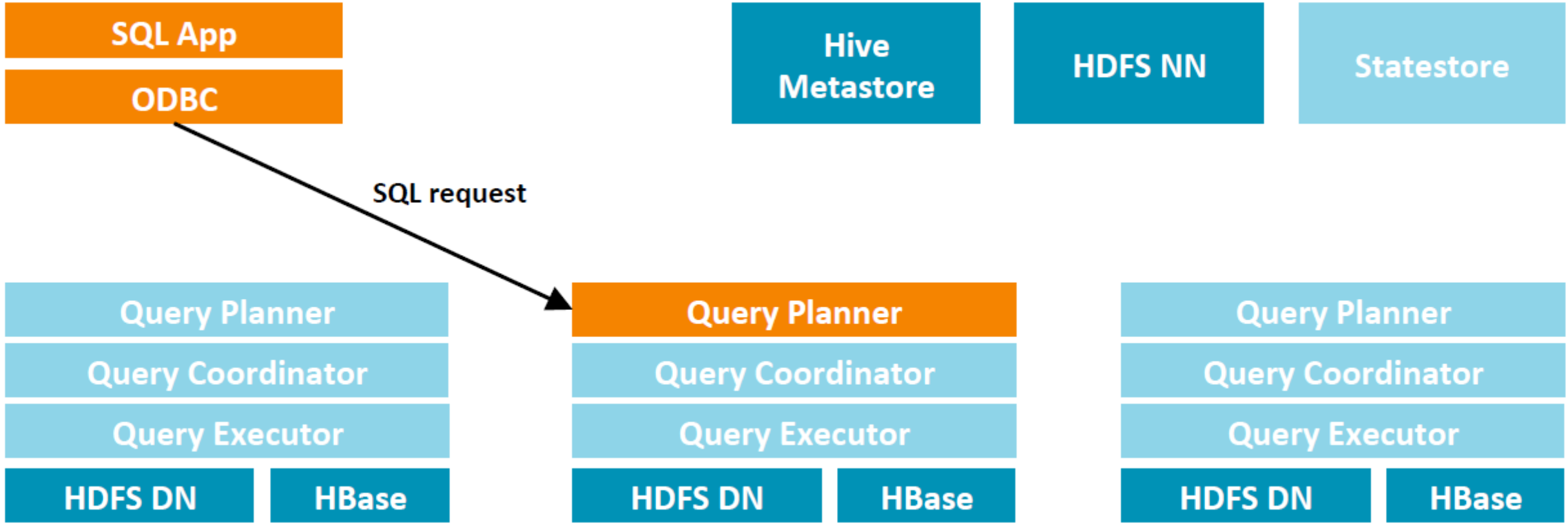
- **Table definitions in shared Hive metastore**
- **Impala tracks additional metadata**
  - *e.g: physical location of blocks in HDFS*
- **After external changes (through Hive or manually to files) metadata needs to be updated**
  - *REFRESH table_name*
  - *INVALIDATE METADATA*

Impala Core Components

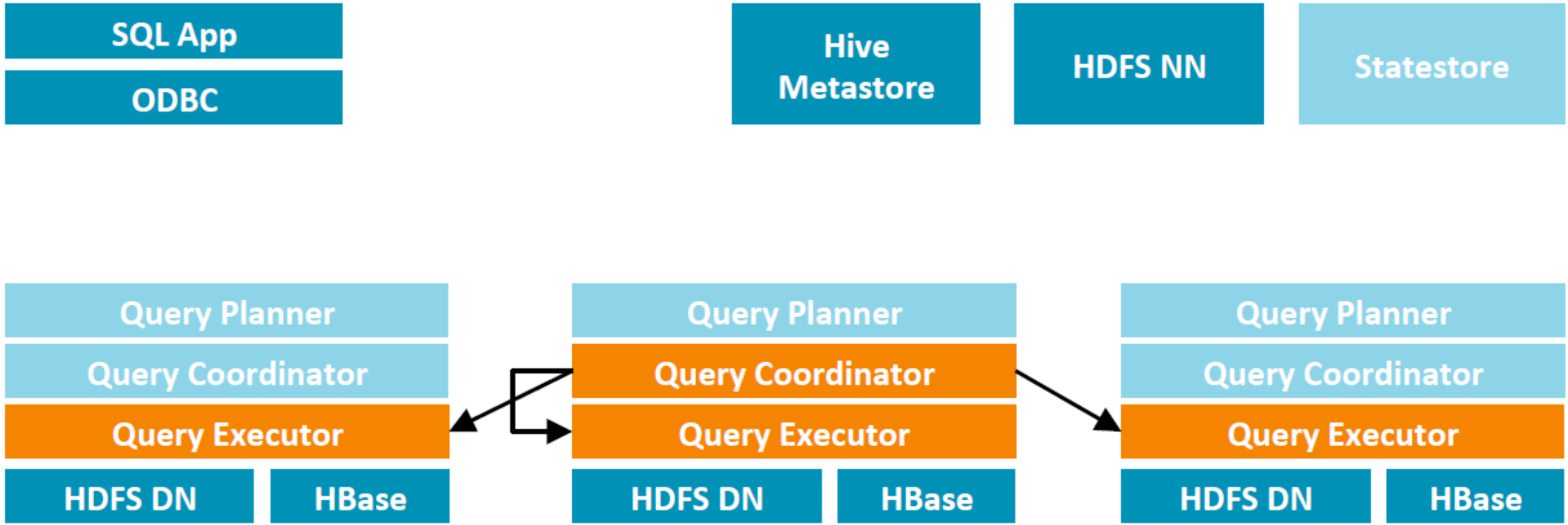
- **impala statestore** (statestored)
  - *One per cluster*
  - *Monitors health of impala daemons*
- **impala catalog service** (catalogd)
  - *One per cluster*
  - *Transfers metadata changes from impala sql statements*
- **impala daemon** (impalad)
  - *One per node*
  - *Accepts queries, distributes work, transfers results back to the coordinator node*



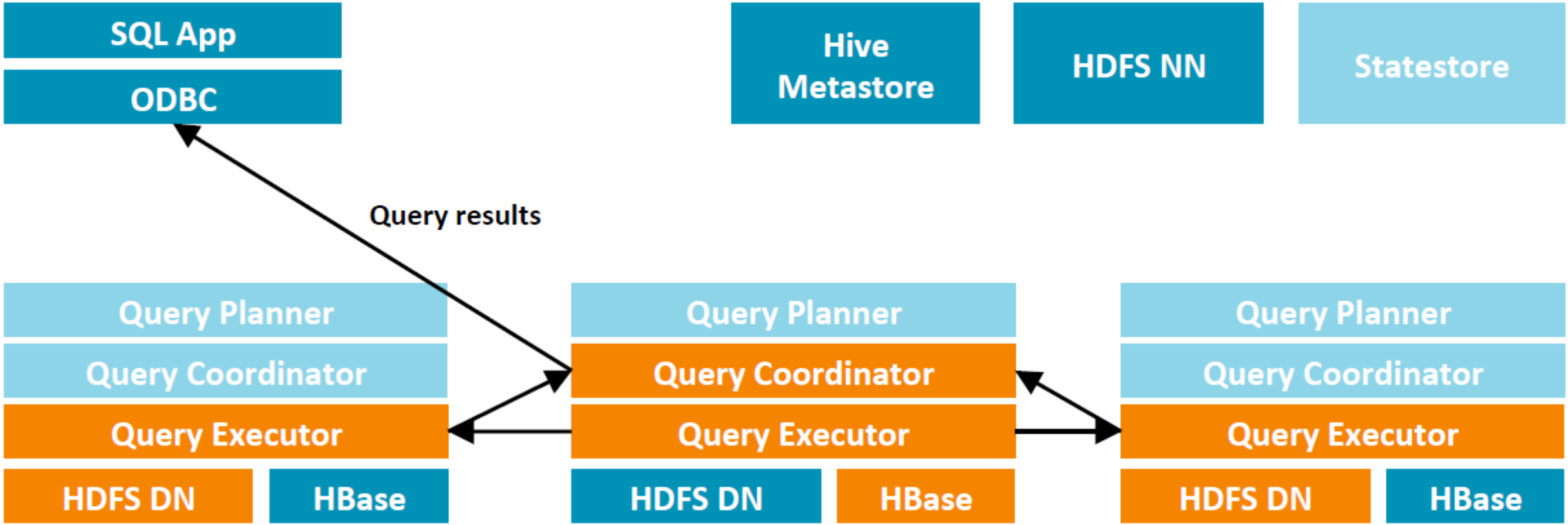
Querying using Impala



Querying using Impala



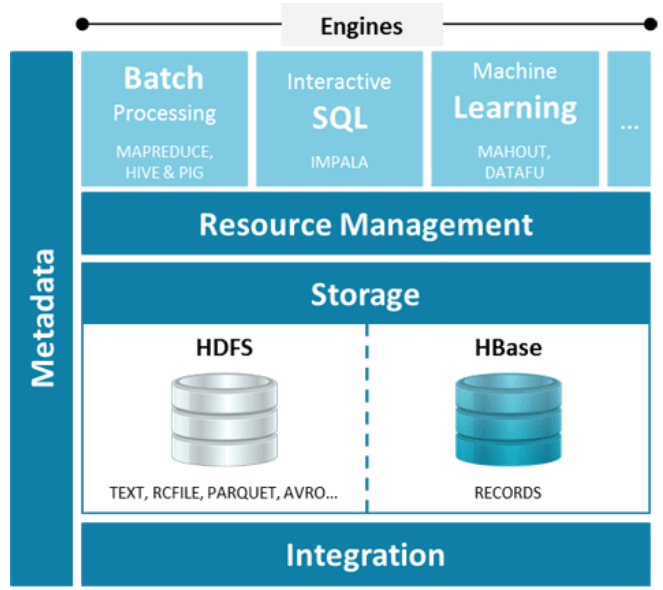
Querying using Impala



Impala vs Hive

- **Hive: MapReduce as an execution engine**
  - *High latency, low throughput queries*
  - *Fault-tolerance model based on MapReduce's*
  - *Easy late-binding of functionality: file formats and UDFs.*
  - *Extensive layering imposes high runtime overhead*

- **Impala: Do not use MapReduce**
  - *direct, process--to--process data exchange*
  - *No fault-tolerance*
  - *An execution engine designed for low runtime overhead*



Starting Impala

- For interactive commands

```
$ impala-shell
```

- To Refresh metadata

```
invalidate metadata;
```

Impala Example

```
select * from students ;
+------------+---------+-------+-----------------------+-----------+
| id         | name    | state | street                | zipcode   |
+------------+---------+-------+-----------------------+-----------+
| student1   | Alice   | CA    | 123 Ballmer Av        | 12345     |
| student2   | Bob     | CA    | 1 Infinite Loop       | 12345     |
| student3   | Frank   | CA    | 435 Walker Ct         | 12345     |
| student4   | Mary    | CA    | 56 Southern Pkwy      | 12345     |
+------------+---------+-------+-----------------------+-----------+
```

Apache HCatalog

Metadata sharing in Hadoop

McGill

FALL 2018

What is Apache HCatalog?

- A Hive metastore interface set
- Shared schema and data types for Hadoop tools
- Assists inter operability between Pig, Hive and Map Reduce
- Table abstraction of data storage
- REST interface for external data access
- Supports reading and writing files in any format for which there is a Hive Serde available.
- Provides data availability notifications

Pig Developers

- To process data using Pig, developers need to know :
  - Where data is located?
  - What is the data format?
  - What is schema to use?

```
songs = LOAD 'log' USING PigStorage(',') AS (artistName, songId, timestamp, user);
artists = FOREACH songs GENERATE artistName;
groupped = GROUP artists BY artistName;
counted = FOREACH groupped GENERATE group AS aritstName, COUNT(artists) AS cnt;
STORE counted INTO 'artist-count-pig';
```

Hive Developers

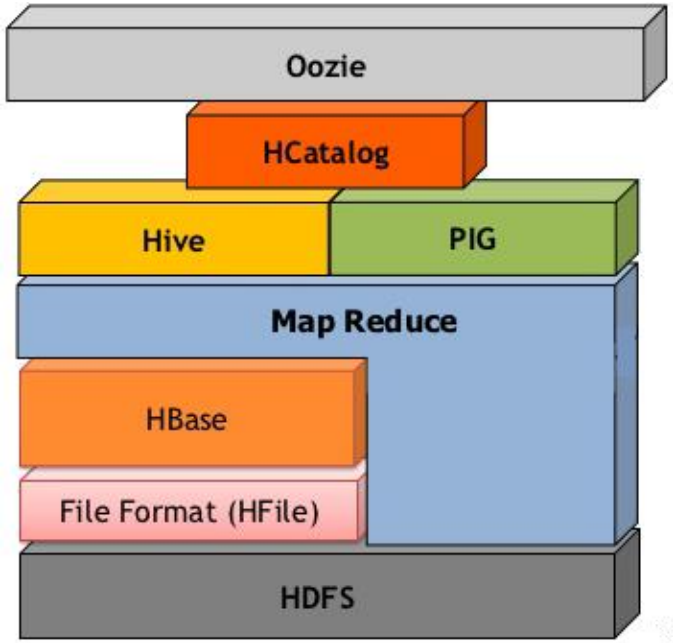
- To analyze data using Hive developers need to know:
  - Where data is located?
  - What is the data format?
  - What is schema to use?
  - How to load data into tables/partitions ?
- Create « External » tables to read data

```
CREATE TABLE `orders_stage`(
  `order_no` int,
  `quantity` string,
  `amount` string,
  `last_update_date` date)
PARTITIONED BY (
  `order_date` date)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '|'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  'hdfs://sandbox:8020/apps/hive/warehouse/orders_stage'
TBLPROPERTIES (
  'transient_lastDdlTime'='1486239288')
```

HCatalog Benefits

- **Abstracts location and format of the data**
  - **Use only the dataset name, rather than the name and the path**
  - **Store datasets definition (and properties) in Hive metastore**
- **Enable data discovery**
  - **Browsing/Inspecting metadata**
  - **Searching datasets**
- **Enable notifications of data availability**

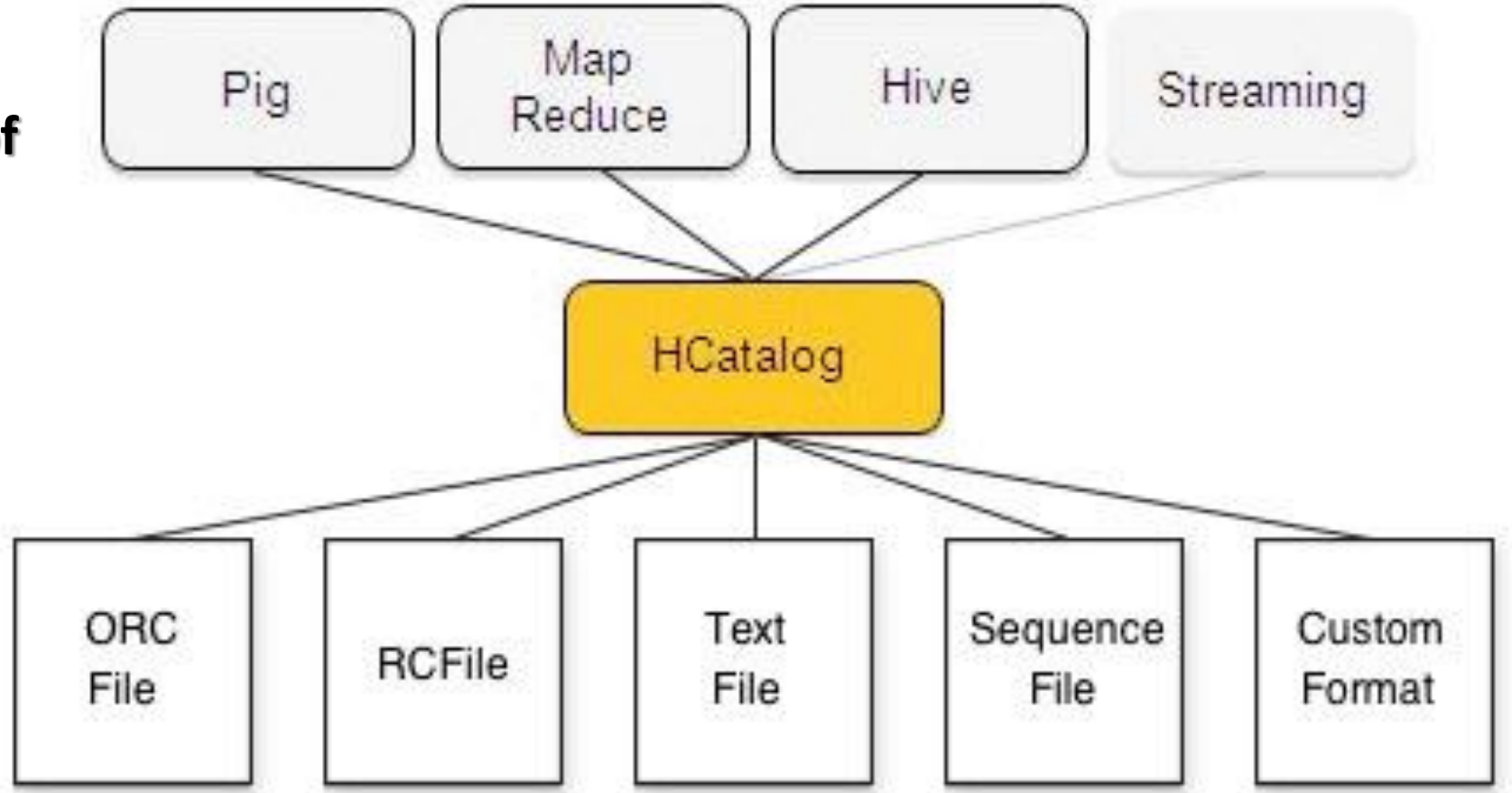
  *HCatalog uses JMS (ActiveMQ) notifications that can be sent for **add/drop** database, table, partition.*



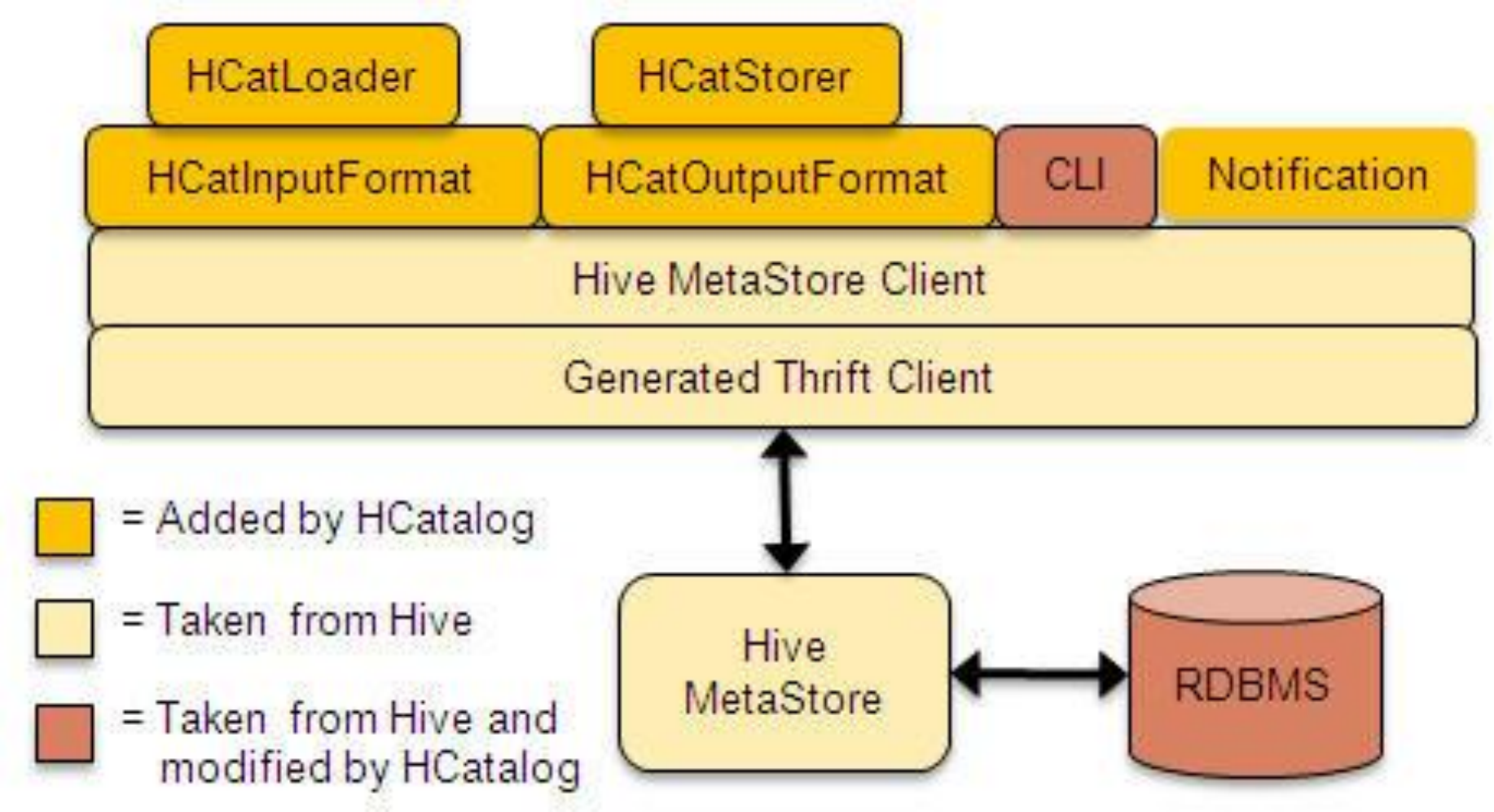
HCatalog : Supported Formats

HCatalog = Table abstraction of data storage

No need to care Where, How, data is stored



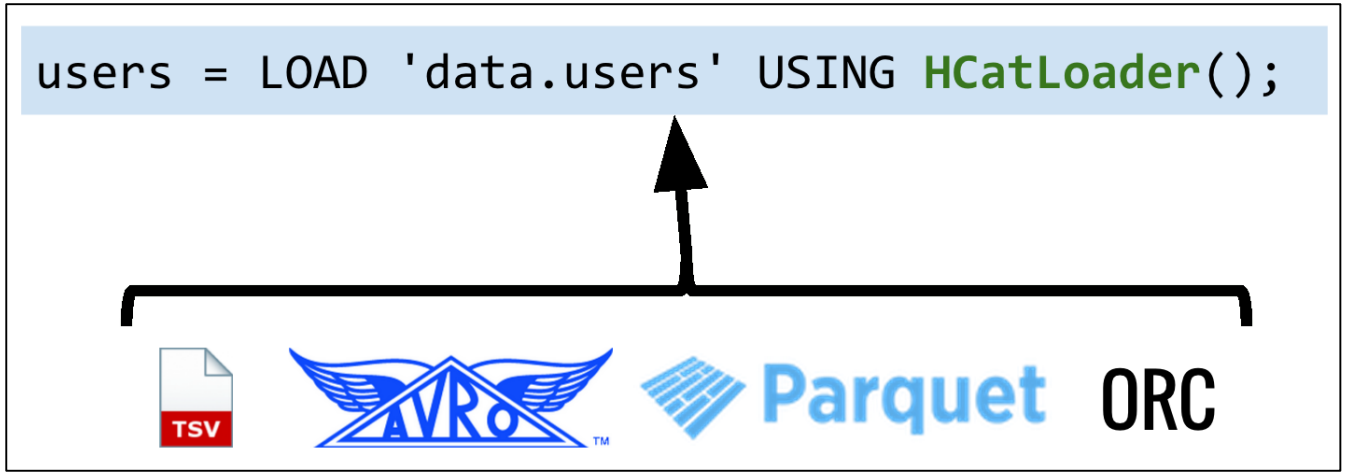
HCatalog High Level Architecture



Using Pig with HCatalog

HCatalog
Table Management

- **HCatLoader** **To Load data**
- **HCatStorer** **To store results**



Example : Reading using HCatLoader()

```
raw = load 'streams' using HCatLoader();
valid = filter raw by date = '20140101' and isValid(duration);
```

Exemple : Writing using HCatStorer ()

```
store valid into 'streams_valid' using HCatStorer()
('date=20110924');
```

Running HCatalog

$ **hcat**

```
[cloudera@quickstart ~]$ hcat
usage: hcat { -e "<query>" | -f "<filepath>" } [ -g "<group>" ] [ -p "<perms>" ] [ -D"<name>=<value>" ]
 -D <property=value>   use hadoop value for given property
 -e <exec>             hcat command given from command line
 -f <file>             hcat commands in file
 -g <group>            group for the db/table specified in CREATE statement
 -h,--help             Print help information
 -p <perms>            permissions for the db/table specified in CREATE statement
[cloudera@quickstart ~]$
```

WebHCat server default port : 50111

HCatalog Example

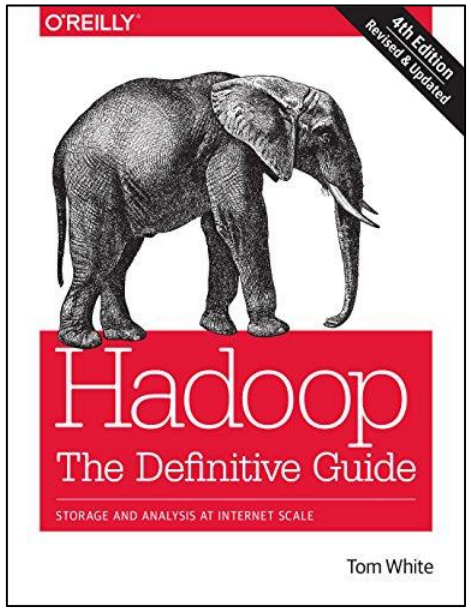

```
# Create a table using the command line

$  hcat -e "create table groupids (name string,id int)"

or

$  hcat -e "create table groups (name string,placeholder
   string,id int) row format delimited fields terminated
   by ':' stored as textfile"


# Get the schema for a table using the command line

$  hcat -e "desc groups"
```

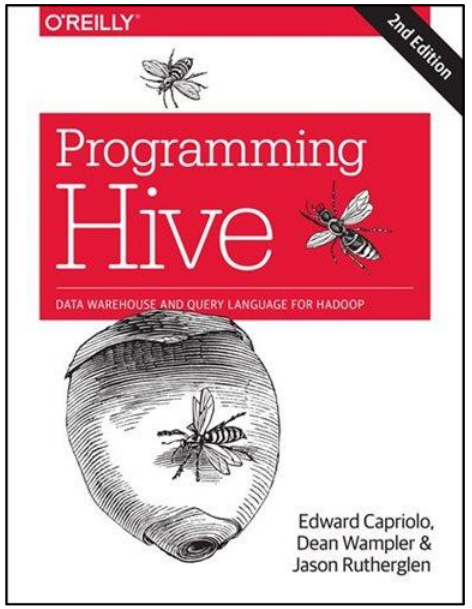
It’s time for a break

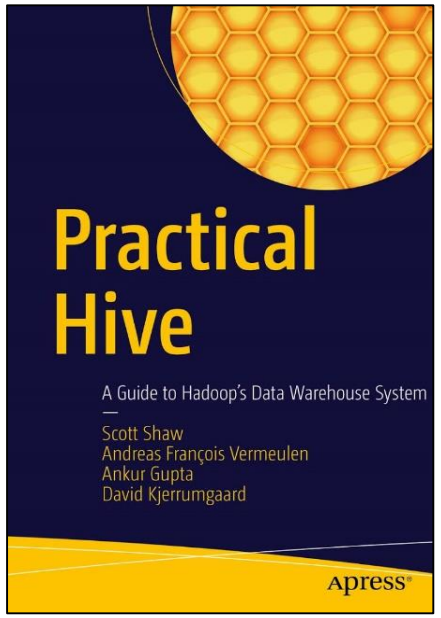
Grab some coffee, We’ll be back in 15min

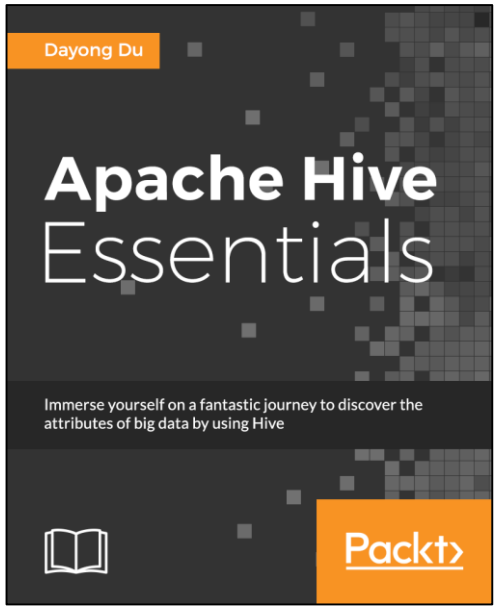
Hive - Impala Workshop

Resources







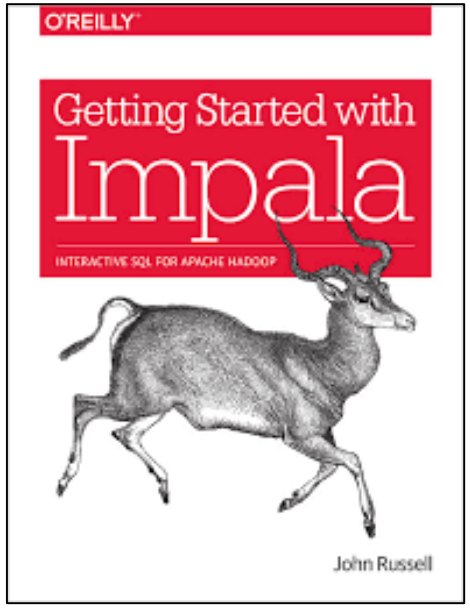


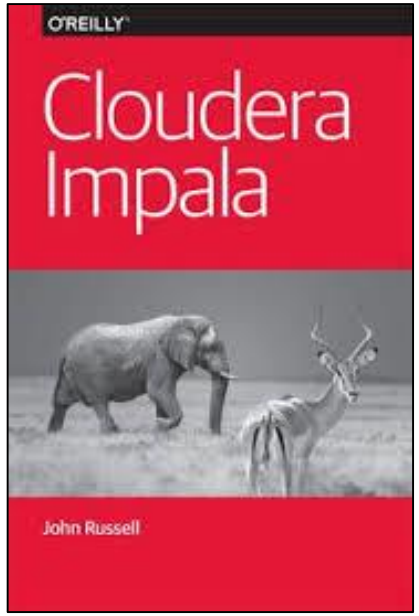
Online Resources

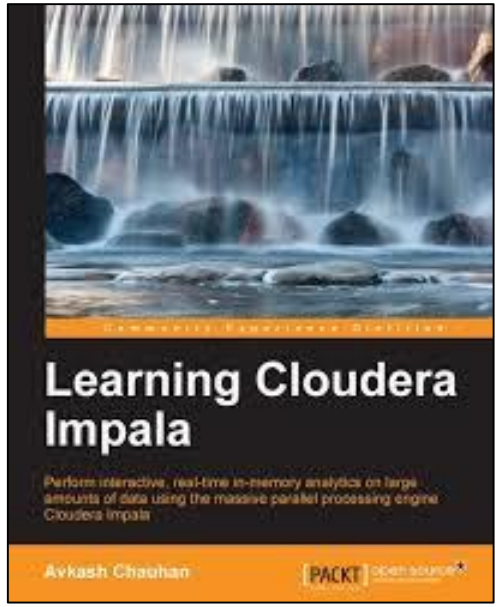
https://hive.apache.org/

https://cwiki.apache.org/confluence/display/Hive/LanguageManual

Resources







Online Resources

https://impala.apache.org/

Thank You