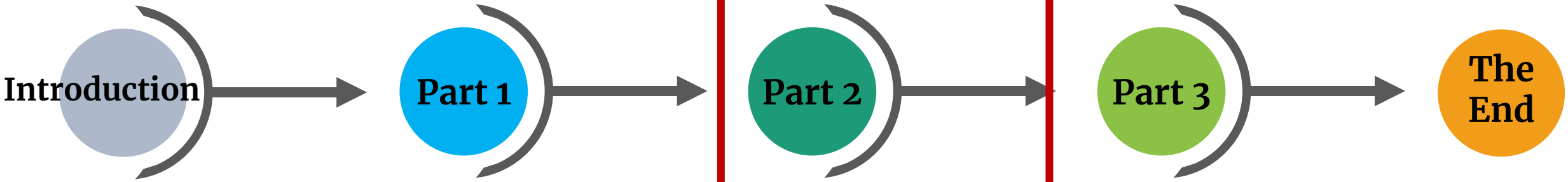
YCBS 257 – Data at Scale

Instructor: **Dr Khaled Tannir**

McGill | School of Continuing Studies

Course Learning Path (*CLP*)



Data at Scale Characteristics

- *Review Big Data 5Vs: Volume, Velocity, Variety, Veracity and Visualization*
- *Introducing Hadoop Ecosystem and Hadoop Distributions*

Data Storage & Processing

- *Learn data characteristics*
- *Learn how to store large volume datasets*
- *Learn to process large volume datasets*

Data Analysis

- *Learn to analyze large volume structured data format*
- *Learn to analyze large volume unstructured data format*
- *File Formats*

Data ingestion

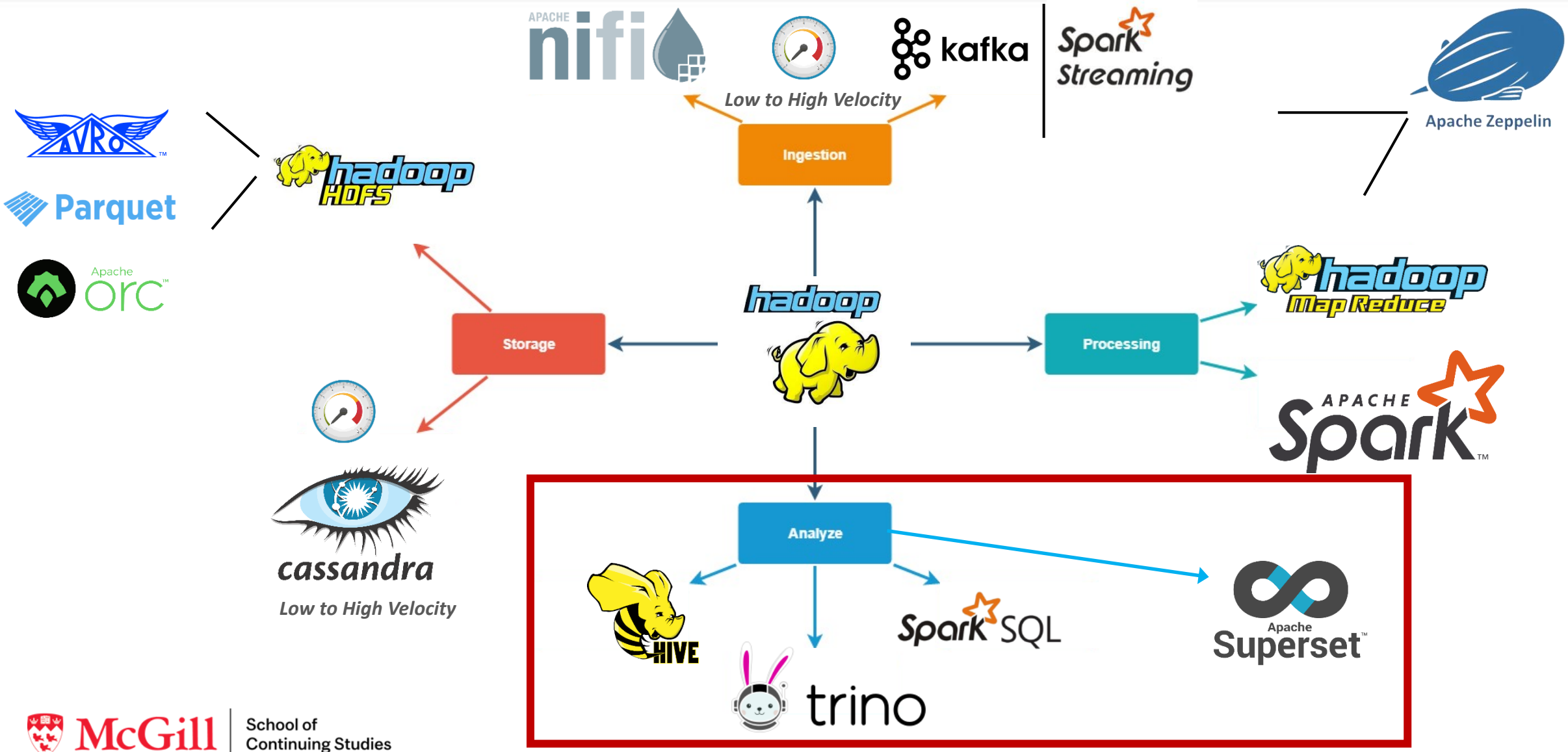
- *Learn to ingest on-demand structured data*
- *Lear to ingest real time structured and unstructured data*
- *Learn to collect events and capture data change (CDC)*

Volume

Veracity
Variety
Visualization

Velocity

Data at Scale Popular Technologies



Data at Scale Course

Part 1

Part 2

Volume

Low Velocity — High Velocity

Distributed Storage

Volume

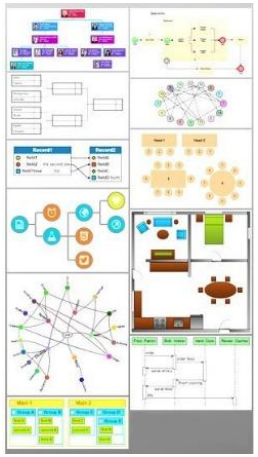
On Disk — In-Memory

Distributed Processing

Variety

Unstructured — Structured

Visualization

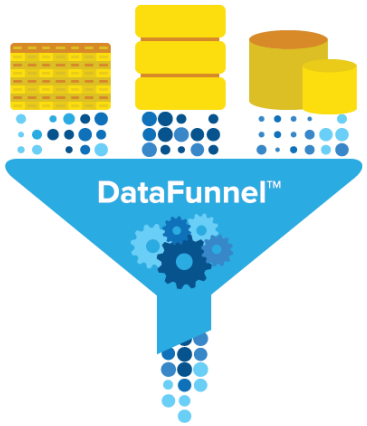


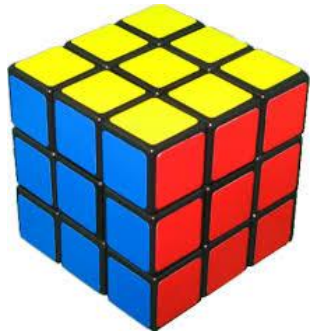
4

McGill School of Continuing Studies

Khaled Tannir

The Big Data Process Model





1 Collect

Collect data from sources

- Open data
- Traditional ETL
- ERP / Data warehouse
- RDBMS
- …

2 Organize

Prepare the data

- Cleansing
- Filtering
- Sorting
- Define the format
- Define the lifecycle.
- ….

3 Analysis

Extract Knowledge from the data

- Find insights
- Modeling behavior
- Probabilistic rather then definitive.
- Text, voice, image analysis
- …

4 Decide

Use insights to change business

- Reports
- Dashboards
- BPM – Real time decision
- Information sharing
- …

Theme of this Course

- ***Analyzing structured Data at Scale Concepts***

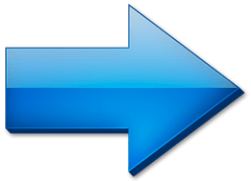
- *Apache HIVE*
  - *Core Concepts*
  - *Hive Architecture*
  - *Hive Data Model*
  - *HiveQL query Language Essentials*

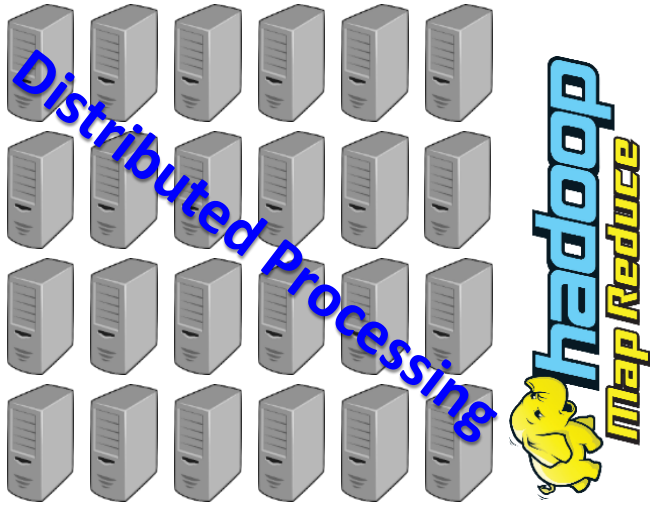
Data at Scale

Large Volume

Low Velocity

Different formats





Structured Data

Spark SQL

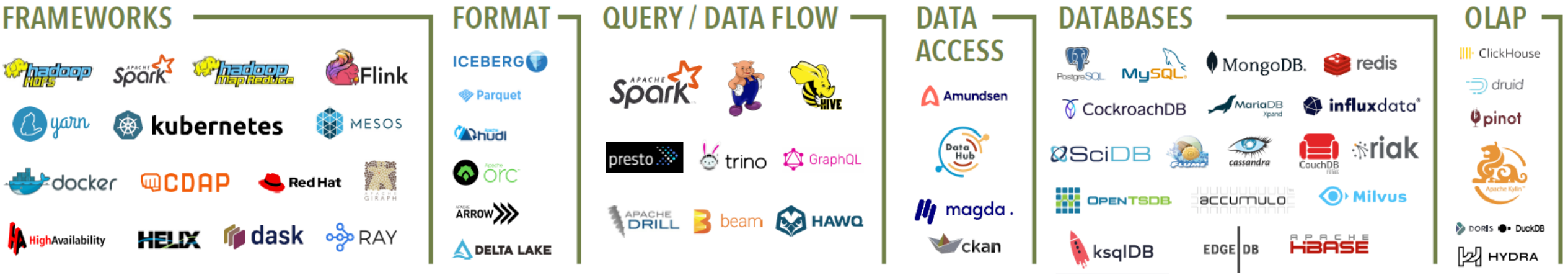
HIVE

trino

Analyze

APACHE Spark™

Popular Tools to Analyze Data at Scale



MAD Landscape 2023 - https://mattturck.com/mad2023/

Analyzing Data at Scale

- ***How to analyze a distributed file / Dataset ?***

  How to organize/query data in a distributed environment?

- ***What if you have to analyze billions of rows every day?***

  How long will take analyzing your dataset?

Analyzing Data at Scale

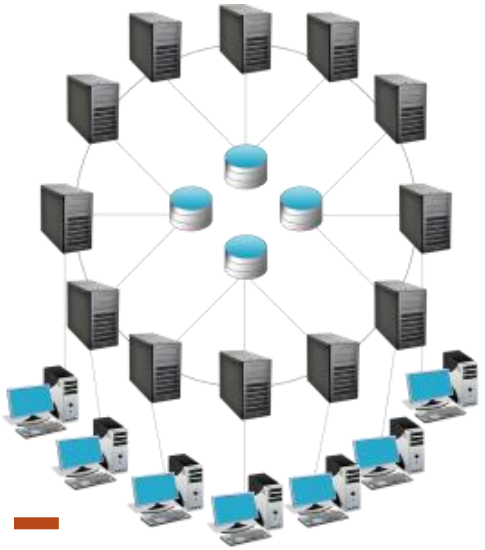
- **To analyze/query large / distributed datasets you need to consider**

1. **Data format (ORC, Avro, Parquet, …)**

   Reduce the volume and Optimize the structure of datasets

2. **Data Partitioning**

   Decomposing datasets into more manageable parts



3. **Data Bucketing (Hive/Spark)**

   Provides a way of segregating equally data into multiple partitions (files/directories)

1- Data Format

- **CSV, JSON, XML, TXT, … Human readable, doesn't mean that's the best way to store data** *(High disk space usage)*
  - **Those file formats cannot be stored in a parallel manner** *(Cannot parallelize data processing)*

- **Avro, ORC, and Parquet Three optimized file formats (***ORC=Optimized Row Columnar***)**
  - **Each of them is unique and bring its own relative advantages and disadvantages**

2- Data Partitioning

- **HDFS does not have the concept of indexes.**
- **Even for reading one row, the entire file should be read.**

  High disk I/O and delays

- **Partitioning provides a way to read only a subset of data.**

  *Based on a Partition **key** (Column Name)*

- **Multiple attributes can be used for hierarchical partitioning.**
- **Split data into directories based on individual values of attributes.**

  *Partition **value** (Column Value)*

3- Data Bucketing

- **Partitioning is only optimal when a given attribute has a small setup unique values**

What if we need to partition for a key with a large number of values without proliferating the number of directories?

- **Bucketing is similar to Partitioning (Even distribution)**
- **It uses a hash function instead of the value of the attribute to convert the value into a special hash key**

Values that have the same hash key go to the same bucket

- **Number of Buckets can be controlled**
- **Ideal for attributes with large number of unique values**

What is Apache Hive?

- A Data Warehouse on top of Hadoop (SQL on Hadoop).
- Can deal with different storage and file formats using *Input/Output* and *SerDes.*
- Summarize Big Data and makes querying and analyzing easy.
- Familiar, scalable, and extensible.
- Written in Java and open-source

Features of HIVE

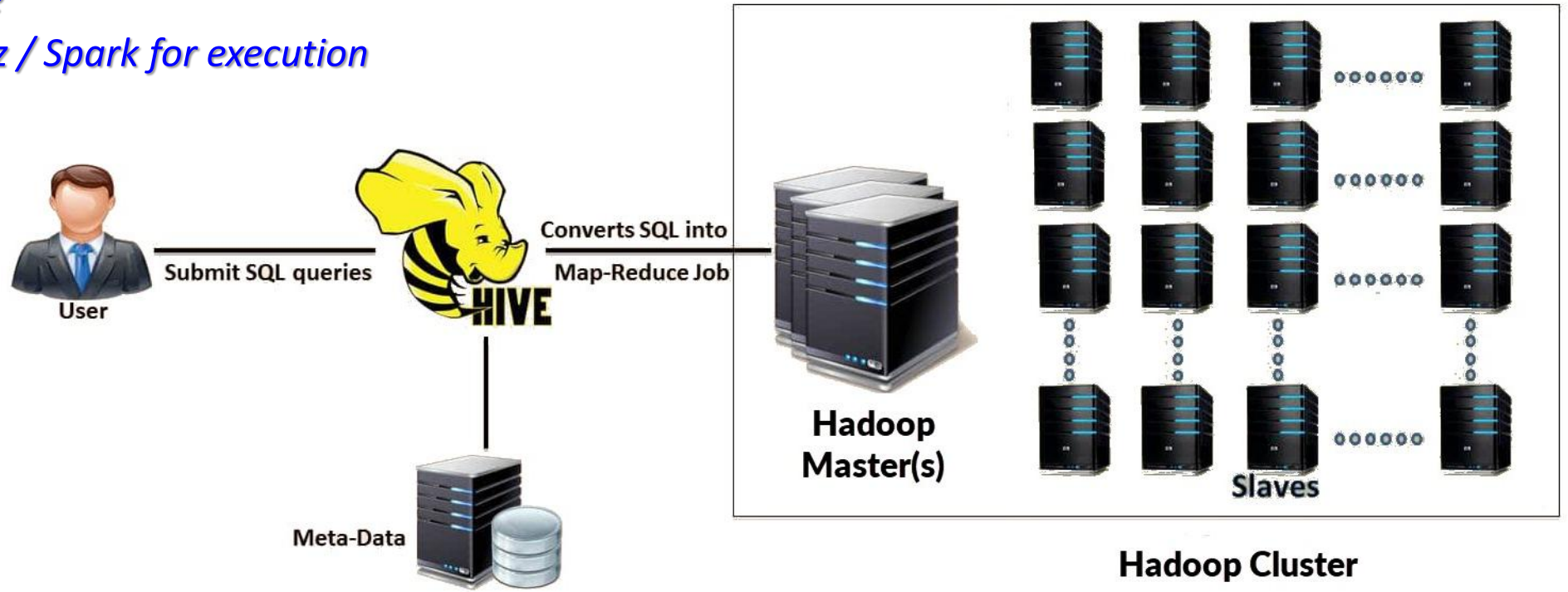
- Stores schemas in a database (metastore) and processes data stored in HDFS.
- Supports User-Defined Functions (*UDF*) , scripts, and a customized I/O format to extend its functionality.
- Supports running on different computing frameworks *(MapReduce, Tez, Spark..)*
- Provides HiveQL (SQL like Query Language) and Rich data types (structs, lists and maps).

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

- Hive is able to query PB of data.

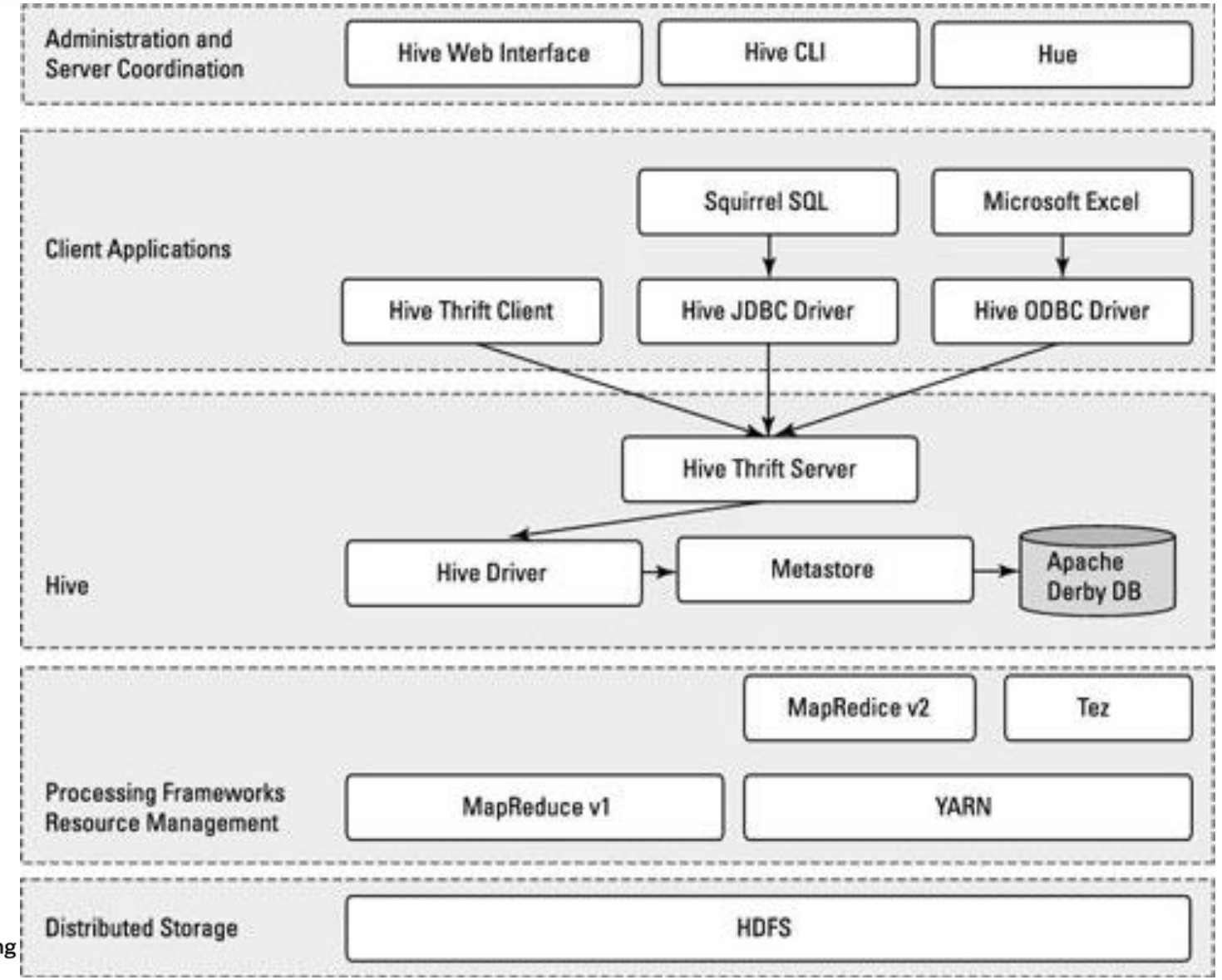
How it Works?

- **Hive is built on top of Hadoop**
  - *Uses HDFS for storage*
  - *Uses MapReduce / Tez / Spark for execution*

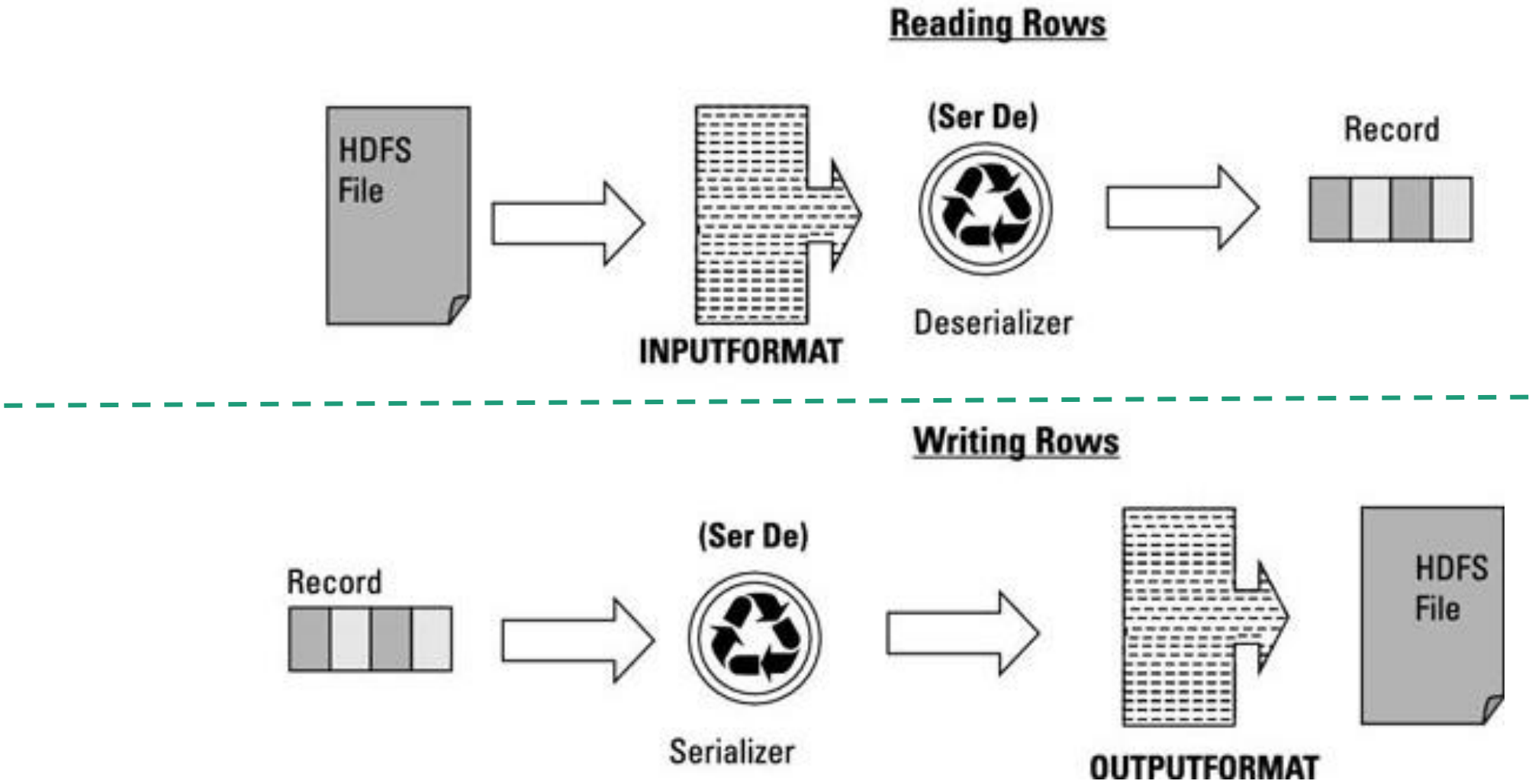


- **Hive compile HiveQL queries into MapReduce *Tez / Spark* jobs and run the jobs in the Hadoop cluster**

Hive High Level Architeture



Hive Read/Write Operations



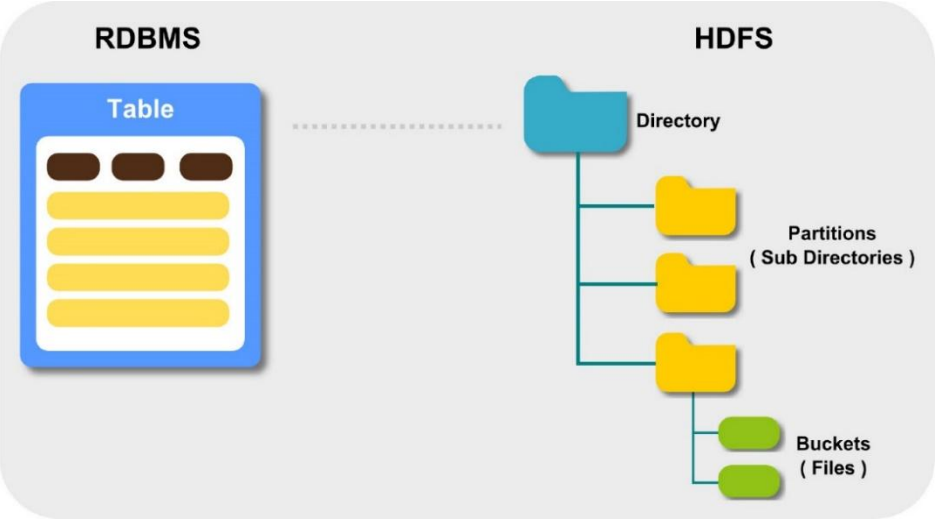
Hive Metastore

- Internally stored in a relational database
- A service that provides metastore access to other Apache Hive services
- Disk storage for the Hive metadata which is separate from HDFS storage.

- Databases
- Tables
- Columns
- Partitions
- Serdes
- Indexes
- Functions
- Statistics
- …

Hive Data Model

- **Database**
  Namespace containing a set of tables
- **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*
  - *Stored in a sub-directory within a table's directory*
- **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*



Default HDFS location:

hdfs:///user/hive/warehouse

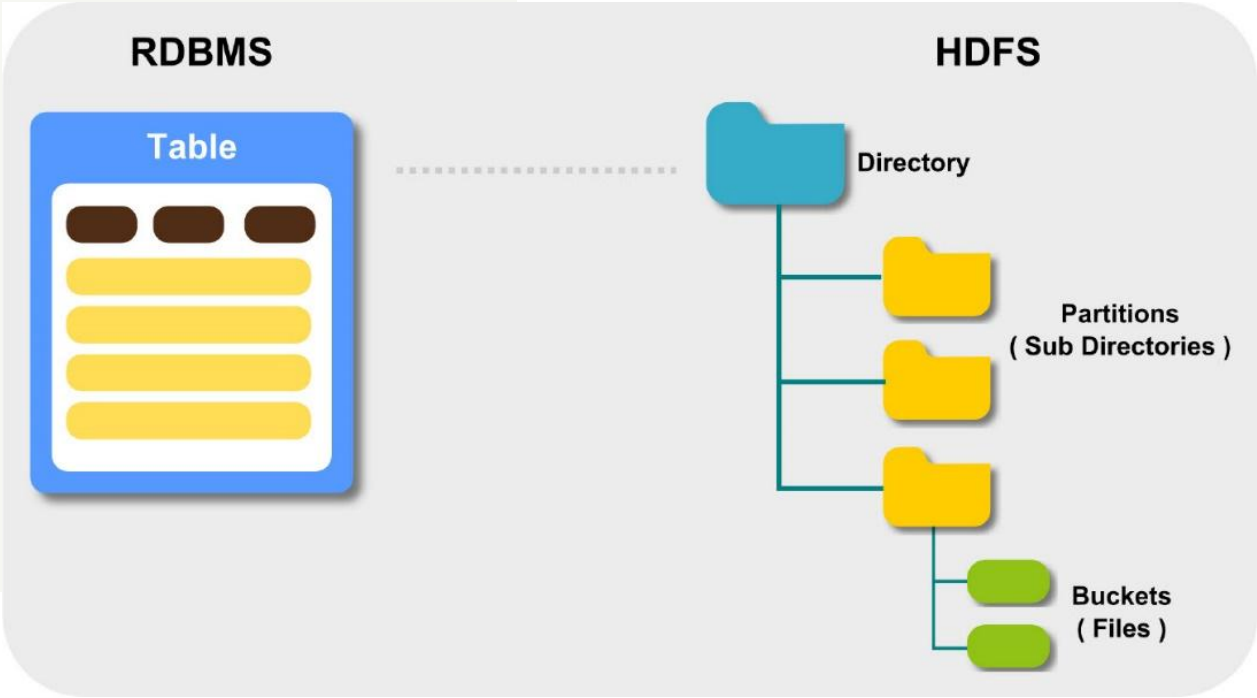
Hive Partitioning Model

- **Key idea – Organizing data in a directory tree**

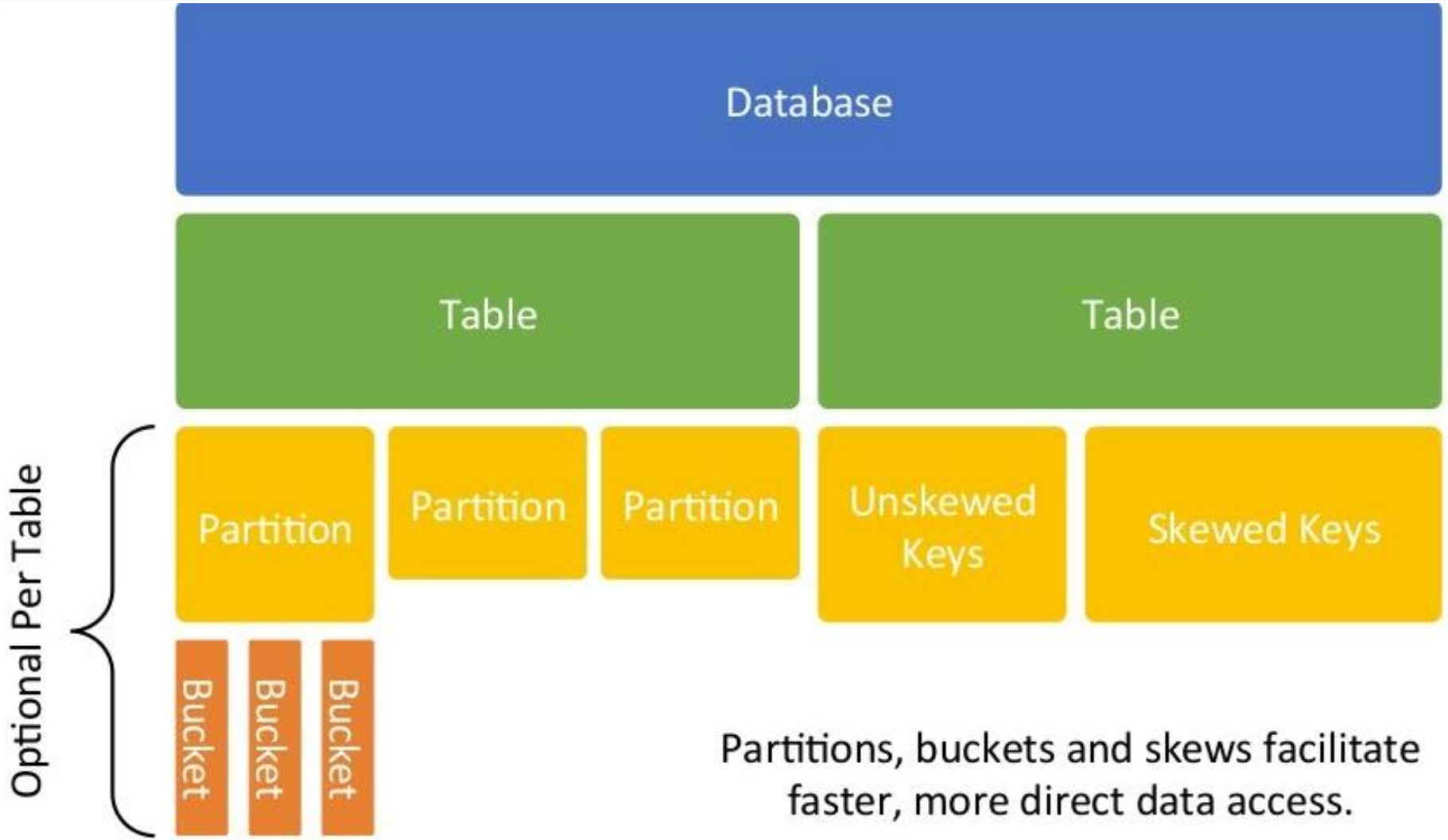
  - Filter by directories as columns

```
SELECT ... WHERE date = '20180513' AND hour = 19
```

```
date=20180513/
 |- hour=18/
 |    |- ...
 |- hour=19/
 |    |- 000000_0
 |    |- ...
 |    |- 000031_0
 |- hour=20/
 |    |- ...
 |- ...
```



Data Abstraction in Hive



** A skewed table is a special type of table where the values that appear very often (heavy skew) are split out into separate files and rest of the values go to some other file.*

Hive Tables Types

Tables

Internal or Hive Managed

Table's schema and data are managed by Hive

External or User Managed

Table's schema is managed by Hive

Table's data is managed by User

Hive Tables Types

Tables

Drop Internal Tables

Will Delete

Schema + Data

Drop External Tables

Will Delete

Schema Only

Hive Tables Types

Internal Tables

External Tables

- **Internal** Table (Hive Managed)
  Table Schema and Data are managed by Hive
- We use **Internal** table when:
  - Data is temporary
  - Data is not needed after table deletion
  - Hive **only** is using the table data
    (no any external sources like pig, sqoop, mapreduce, … use the table)

- **External** Table (User Managed)
  Table Schema is managed by Hive, Data by the User
- We use **External** table when:
  - When data is available in HDFS
  - When files are being used **outside** of Hive

HIVE Managed Tables (*Internal*)

1. **Create the table and its schema**

```
create table products (id int, name string,  price float, category string);
```

2. **Need to Load the data using the Load / Insert into commands**

```
load data inpath /tmp/products.tsv into table products;
```

** products.tsv file is stored on HDFS*

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

USER Managed Tables (*External*)

A. The data is already available on HDFS
   1. Create the table and its schema
   2. Provide a location on HDFS (directory)

```
create external table products (id int, name string,  price float, category string)
location '/tmp/';
```

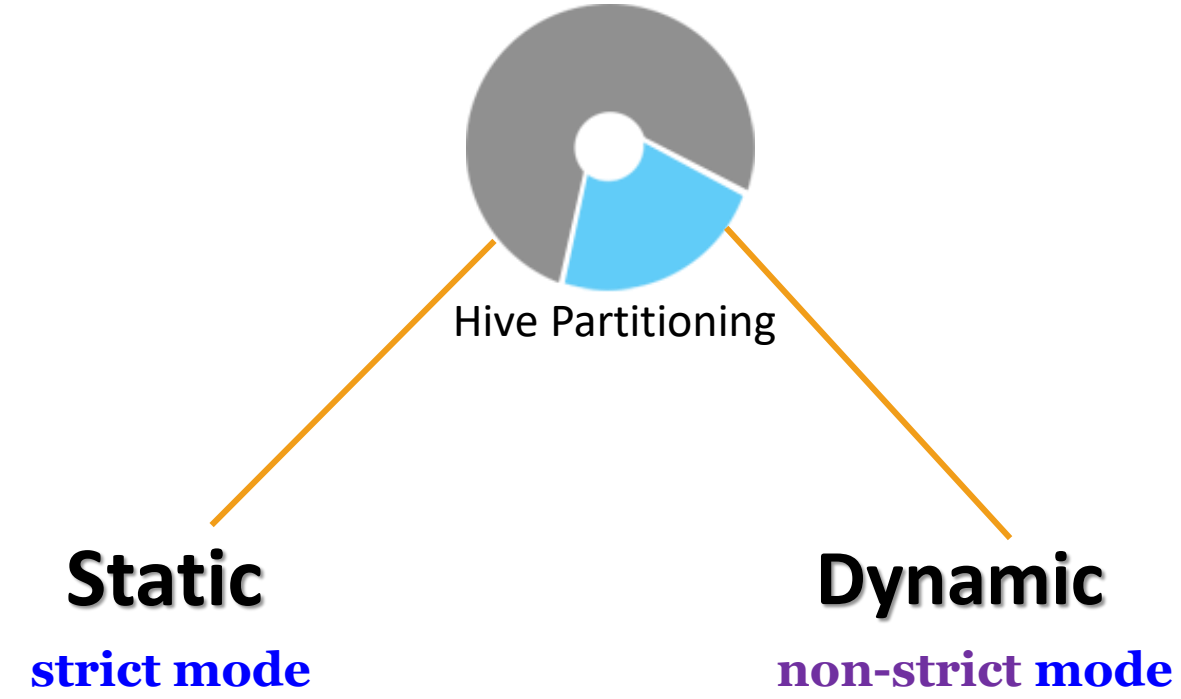
** products.tsv file is stored on HDFS in the /tmp/ directory*

B. Do NOT need a 'LOAD' command to load data

- Only schema will be removed if the table is dropped

Hive Table Partitioning

- Can be performed on Hive Managed and Non-Managed tables
- Static partitions are faster than dynamic partitions
- Strict mode is a useful option in Hive to improve performance and avoid any unexpected delays due to full table scan.

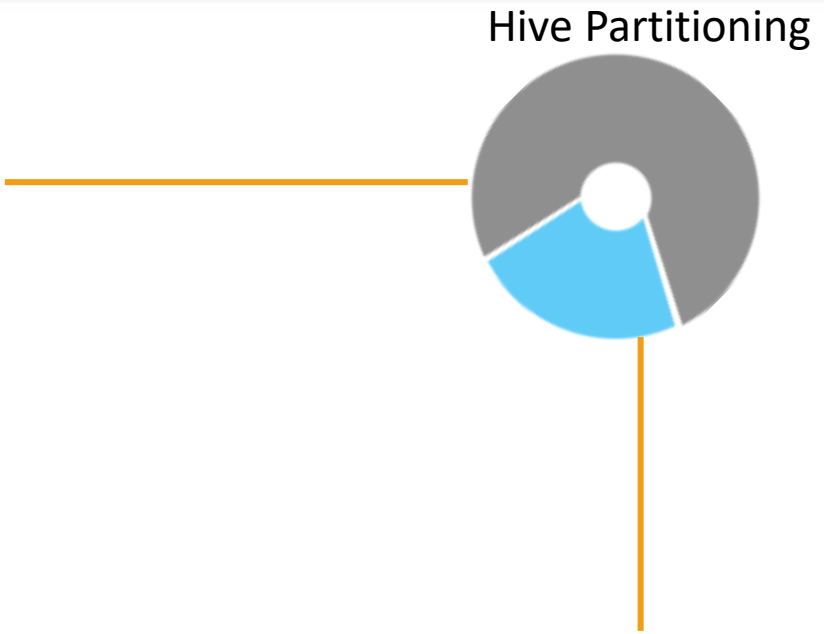



```
set hive.exec.dynamic.partition = true/false;
set hive.exec.dynamic.partition.mode = nonstrict/strict;
```

Hive Table Partitioning

Static Partitioning

- Are preferred when loading big files into Hive tables
- Static partition can be altered
- You “statically” add a partition in the table and move the file into the partition of the table
- You should use where clause to use limit in the static partition.
- Partition column value can be obtained from the filename, day of date etc without reading the whole big file



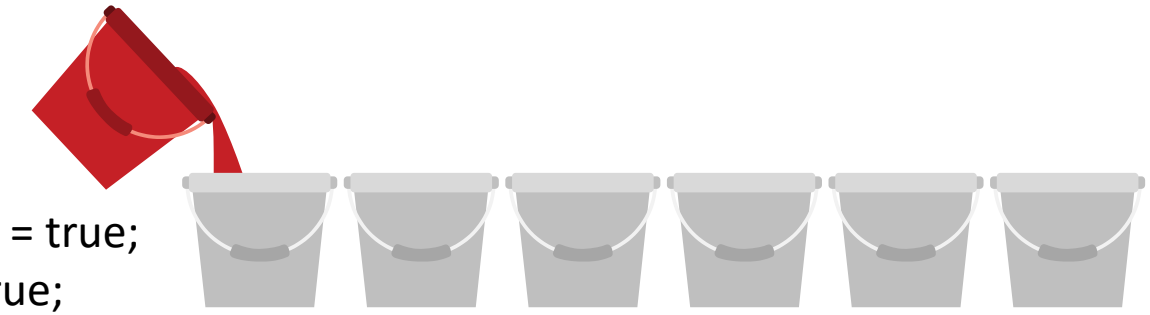
Dynamic Partitioning

- Single insert to partition table is known as a dynamic partition
- Is used loads the data from the non-partitioned table
- Dynamic partition cannot be altered
- Where clause is not required to use limit

Hive Table Bucketing

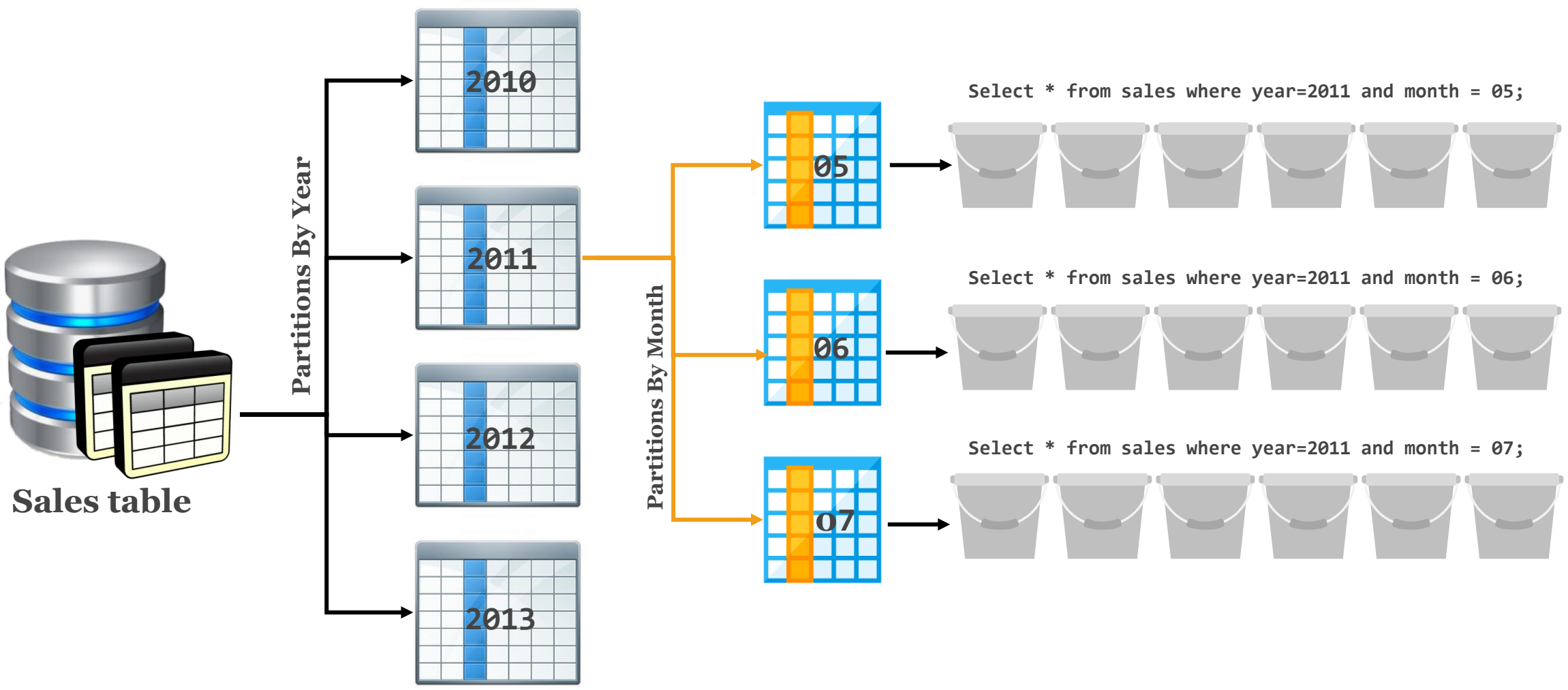
- A technique for decomposing table datasets into more manageable parts
- Based on hashing function on the bucketed column

  Along with mod (by the total number of buckets)
- Create almost equally distributed data file parts
- To divide the table into buckets, we use **CLUSTERED BY** clause
- Each bucket is just a file in the table directory. Bucket numbering is 1-based
- Can be done w/without partitioning



```
set hive.enforce.bucketing = true;
set hive.enforce.sorting=true;
```

Hive Table Partitioning Example



Hive Persistence Formats

- **Built-in Formats:**
  - *ORC File, RCFile (Optimized Row Columnar, Row Columnar)*
  - *Delimited Text*
  - *Avro*
  - *Parquet*
  - *Sequence File*
  - …

- **3rd-Party Addons (need custom Serde):**
  - *JSON*
  - *XML*
  - …

Loading Data in Hive

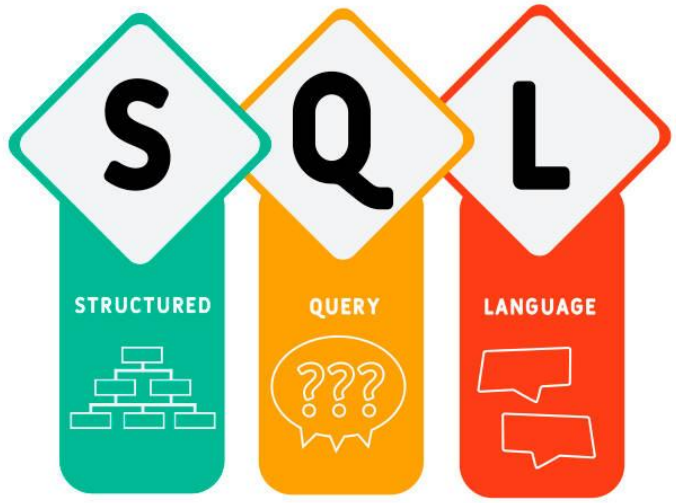
- **Hive LOAD statement** *(Not needed for External Hive Tables)*
  - *Load files from HDFS or local file system.*
  - *Format must agree with table format.*

- **Insert from query**
  - *CREATE TABLE AS SELECT or INSERT INTO.*

- **Spark / Nifi / …**
  - *Data transfer from external sources to HDFS / Hive.*

Hive Query Language (HiveQL)

- **HiveQL / HQL provides the basic SQL-like operations:**
  - **Select columns using SELECT**
  - **Filter rows using WHERE**
  - **JOIN between tables**
  - **Evaluate aggregates using GROUP BY**
  - **Store query results into another table**
  - **Download results to a local directory (i.e., export from HDFS)**
  - **Manage tables and queries with CREATE, DROP, and ALTER**



Primitive Data Types

Type	Comments
`TINYINT, SMALLINT, INT, BIGINT`	1, 2, 4 and 8-byte integers
`BOOLEAN`	TRUE/FALSE
`FLOAT, DOUBLE`	Single and double precision real numbers
`STRING`	Character string
`TIMESTAMP`	Unix-epoch offset *or* datetime string
`DECIMAL`	Arbitrary-precision decimal
`BINARY`	Opaque; ignore these bytes

Complex Data Types

Type	Comments
STRUCT	A collection of elements If S is of type STRUCT {a INT, b INT}: S.a returns element a
MAP	Key-value tuple If M is a map from 'group' to GID: M['group'] returns value of GID
ARRAY	Indexed list If A is an array of elements ['a','b','c']: A[0] returns 'a'

HiveQL : Examples 1/2

- 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 2/2

Hive Managed Table

```
create table products (
        id int,
        name string,
        price float,
        category string)
row format
delimited fields
terminated by ',' ;
```

User Managed Table

```
create EXTERNAL table IF NOT EXISTS Cars (
        name STRING,
        miles_per_Gallon INT,
        cylinders INT,
        horsepower INT,
        Weight_in_lbs INT,
        acceleration DECIMAL,
        `year` DATE)
    COMMENT 'Data about cars from a public database'
    ROW FORMAT DELIMITED FIELDS
    TERMINATED BY ','    STORED AS TEXTFILE
Location '/user/username/visdata';
```

HiveQL : Load / Export Data

Load data into a partitioned table

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

Export data to a file

```
INSERT OVERWRITE DIRECTORY '/tmp/pv_gender_sum'
SELECT  *  FROM pv_gender_sum;
```

Hive SerDes

- SerDe is short for Serializer/Deserializer
- Hive uses the SerDe interface for IO
- The interface handles both serialization and deserialization and interprets the results of serialization as individual fields for processing
- A SerDe allows Hive to read in data from a table, and write it back out to HDFS in any custom format
- Built-in and Custom SerDes

Anyone can write their own SerDe for their own data formats

Connecting to Hive Server

- **Remotely**
  - Zeppelin interface
  - beeline (via JDBC, HiveServer2 support)

    beeline -u jdbc:hive2://localhost:10000
  - JDBC and ODBC drivers *(e.g Squirrel, Tableau, MS Excel…)*
  - Thrift API

- **Run a Hive script**

```
$ hive  -f  <path>/mysql.hql
```

It’s time for a break

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

McGill | School of Continuing Studies

Hive Workshop

McGill | School of Continuing Studies

Hive Query Language

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

- Create table
- Create view
- Drop table
- Load
- …

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
- . . .

Create Table Syntax

General Syntax:

```
CREATE TABLE table_name (
                col1 data_type,
                col2 data_type,
                col3 data_type,
                col4 data_type)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ‘<char>’
STORED AS format_type;
```

Example:

```
CREATE TABLE page_view (
                viewTime INT,
                userid BIGINT,
                page_url STRING,
                referrer_url STRING,
        ip STRING COMMENT 'IP Address of the User’ )
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
   STORED AS TEXTFILE;
```

More Complex Table Example

```
CREATE TABLE employees (
    name STRING,
    salary FLOAT,
    subordinates ARRAY<STRING>,
    deductions MAP<STRING, FLOAT>,
    address STRUCT<street:STRING,
            city:STRING,
            state:STRING,
            zip:INT>)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
  STORED AS TEXTFILE;
```

External Table

```
CREATE EXTERNAL TABLE page_view_stg (
     viewTime INT,
     userid BIGINT,
     page_url STRING,
     referrer_url STRING,
     ip STRING COMMENT 'IP Address of the User’)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t’
STORED AS TEXTFILE
LOCATION '/user/staging/page_view';
```

More About Tables

- **CREATE TABLE**
  - **LOAD:** data file moved into Hive's data warehouse directory
  - **DROP:** both metadata and data deleted

- **CREATE EXTERNAL TABLE**
  - **No LOAD and no files moved**
  - **DROP: only metadata deleted**
  - **Use this when sharing with other Hadoop applications, or when you want to use multiple schemas on the same data**

More about Partitioning

- Can make some queries faster
- Divide data based on partition column
- Use PARTITION BY clause when creating table
- Use PARTITION clause when loading data
- Use MSCK to add missing partitions
- SHOW PARTITIONS will show a table's partitions

More about Bucketing

- **Can speed up queries that involve sampling the data**

  Sampling allows users to take a subset of dataset and analyze it, without having to analyze the entire data set. If a representative sample is used, then a query can return meaningful results as well as finish quicker and consume fewer compute resources.

- **Sampling works without bucketing, but Hive has to scan the entire dataset**
- **Use CLUSTERED BY when creating table**
- **For sorted buckets, add SORTED BY**
- **To query a sample of your data, use TABLESAMPLE**

HiveQL : Sampling

Hive offers a built-in TABLESAMPLE clause that allows you to sample your tables.

TABLESAMPLE can return

- *Only subsets of buckets (bucket sampling)*
- *HDFS blocks (block sampling)*
- *Only first N records from each input split.*

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

Browsing Tables And Partitions

Command	Comments
`SHOW TABLES;`	Show all the tables in the database
`SHOW TABLES 'page.*';`	Show tables matching the specification (uses regex syntax)
`SHOW PARTITIONS page_view;`	Show the partitions of the page_view table
`DESCRIBE page_view;`	List columns of the table
`DESCRIBE EXTENDED page_view;`	More information on columns (useful only for debugging)
`DESCRIBE page_view PARTITION (ds='2008-10-31');`	List information about a partition

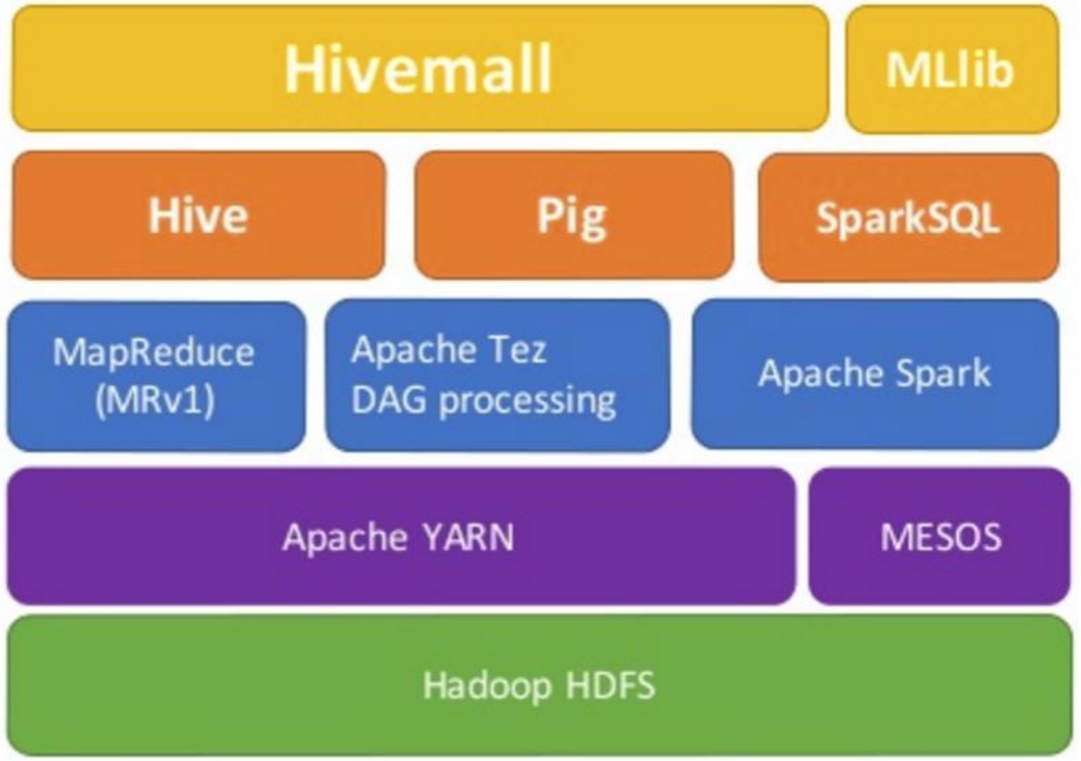
Questions?

McGill | School of Continuing Studies

Appendix

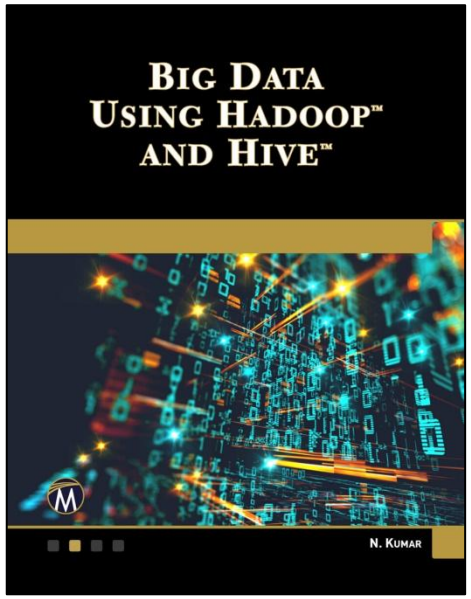
Apache HiveMall

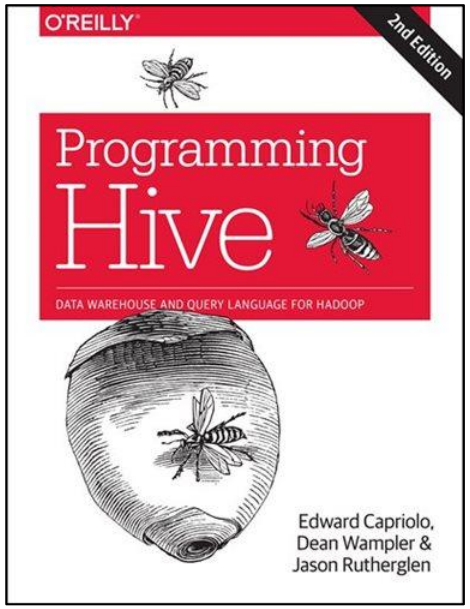
- A scalable machine learning library built as a collections of Hive UDFs.
- Supported Algorithms:
  - Binary Classification
  - Multi-class Classification
  - Regression
  - Recommendation
  - k-Nearest Neighbor
  - Anomaly Detection
  - Natural Language Processing (English/Japanese)

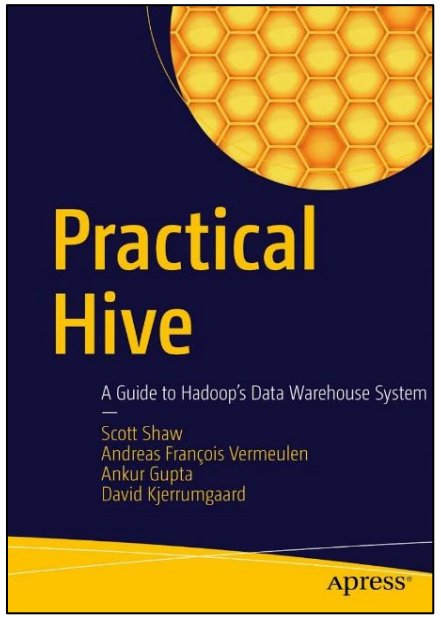


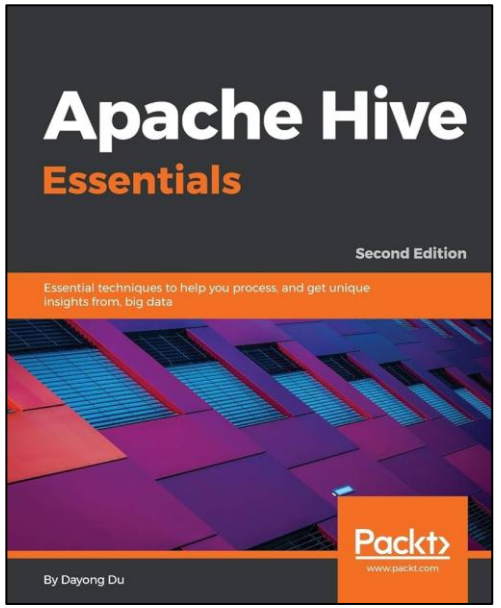
https://hivemall.incubator.apache.org/

Resources









Online Resources

https://hive.apache.org/

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

Merci!

Thank You!

McGill | School of Continuing Studies