HIVE

Outline

- Introduction to Hive
- Hive Architecture
- Hive Data Types
- Hive Tables
- HiveQL
- Hive UDFs

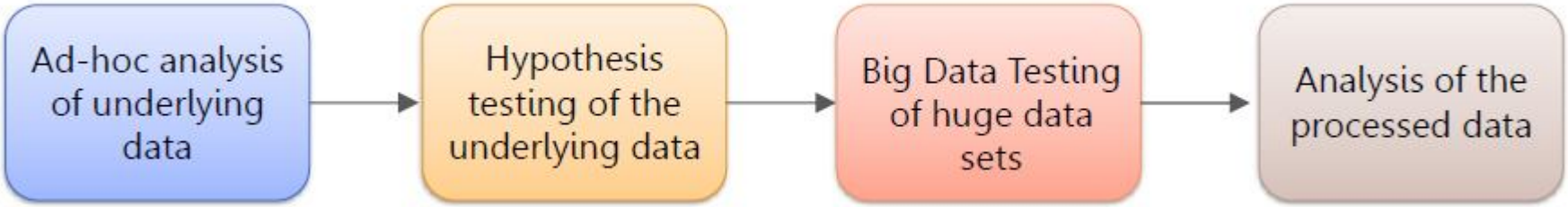
Hive Background

- Hive originated as an internal project in Facebook
- Later it was adopted in Apache as an open source project
- Facebook deals with massive amount of data (petabytes scale) and it needs to perform more than 75k ad-hoc queries on this massive amount of data
- Since the data is collected from multiple servers and is of diverse nature, any RDBMS system could not fit as probable solution
- Map Reduce could be a natural choice, but it had its own limitations

What is Hive?

- It is a query engine wrapper built on top of Map Reduce
- It is treated as Data Warehousing tool of Hadoop Ecosystem
- It is used for data analysis
- Primarily targeted to the users with SQL background
- Provides HiveQL, which is very similar to SQL
- It is used for managing and querying structured data
- Hadoop complexity is hidden from end users
- Java and Hadoop API knowledge is optional for core users
- Developed by Facebook and contributed to community

Hive Use Cases



Hive vs Pig

Pig	Hive
Procedural syntax	Declarative syntax
Tool of choice for programmers	Tool of choice for analysts
Partitions NOT supported	Partitions supported
Doesn't have a server	Has Thrift (optional) server
Pig doesn't provide web interface	Hive provides optional web interface
Pig doesn't support JDBC/ODBC connectivity	Limited support
Suitable for data factory operations	Suitable for data warehousing operations

Hive vs RDBMS

Hive functionality seems quite near to a RDBMS, but there are subtle differences like:

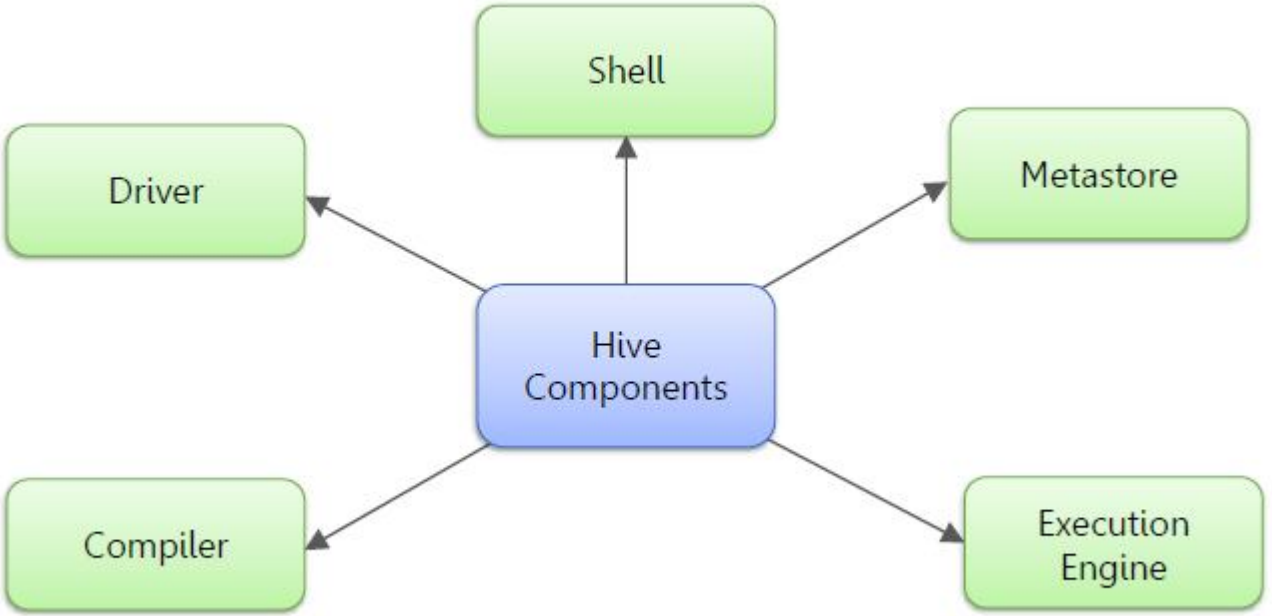
- Data in the RDBMS tables could be updated and deleted selectively, whereas Hive doesn't support it
- Hive is Schema-on-Read system, whereas RDBMS systems are Schema-on-write systems
- RDBMS tables are best suited for small to middle scale volume data (Early GBs max), whereas underlying data in HDFS tables typically of the order of several GBs to TB scale

Limitations of Hive

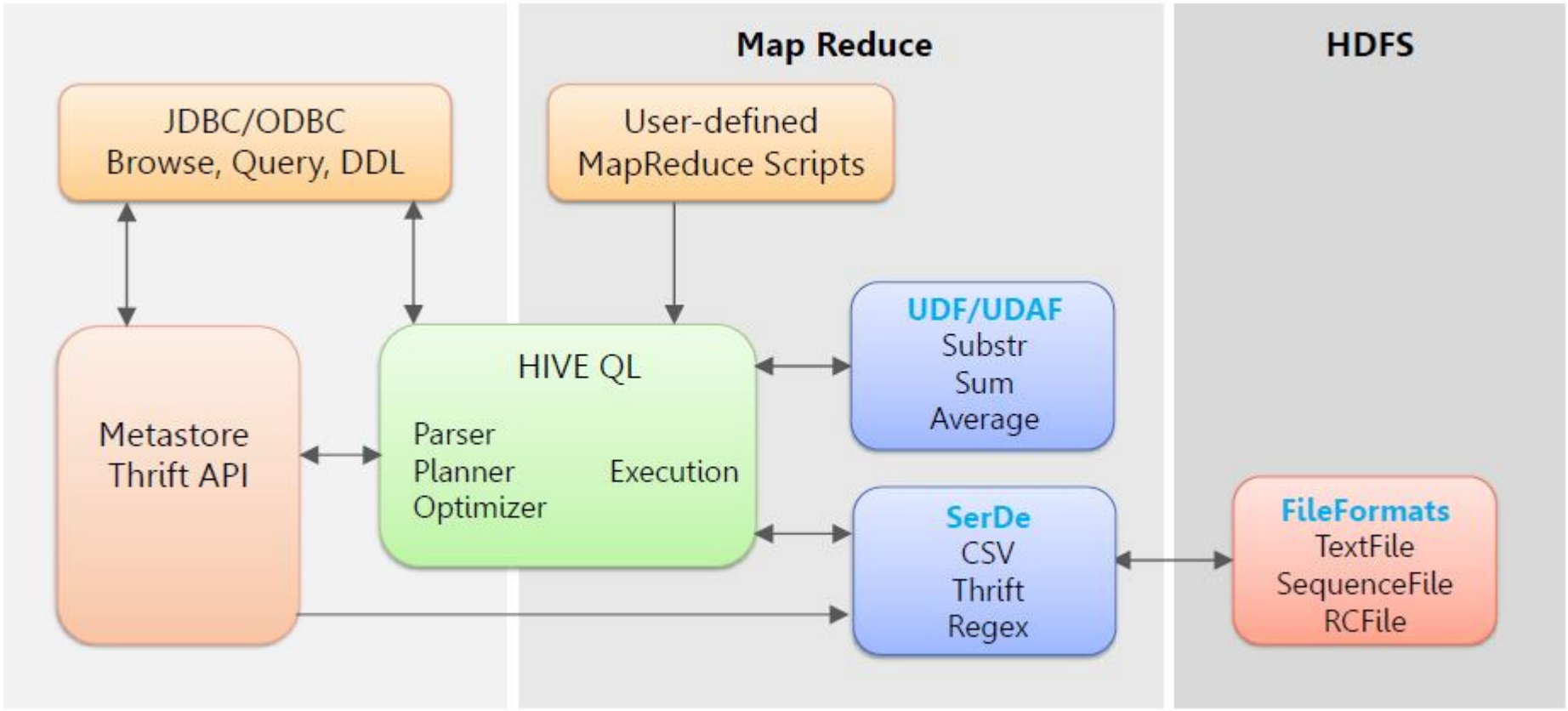
Hive is an excellent query engine on top of Hadoop but:

- Hive typically suits for high latency queries. The response of fastest Hive query is in the order of several seconds
- Hive does not support row level updates/deletes. Data can be either overwritten or appended
- Hive can not replace an OLTP system
- Hive does not offer real-time queries capability

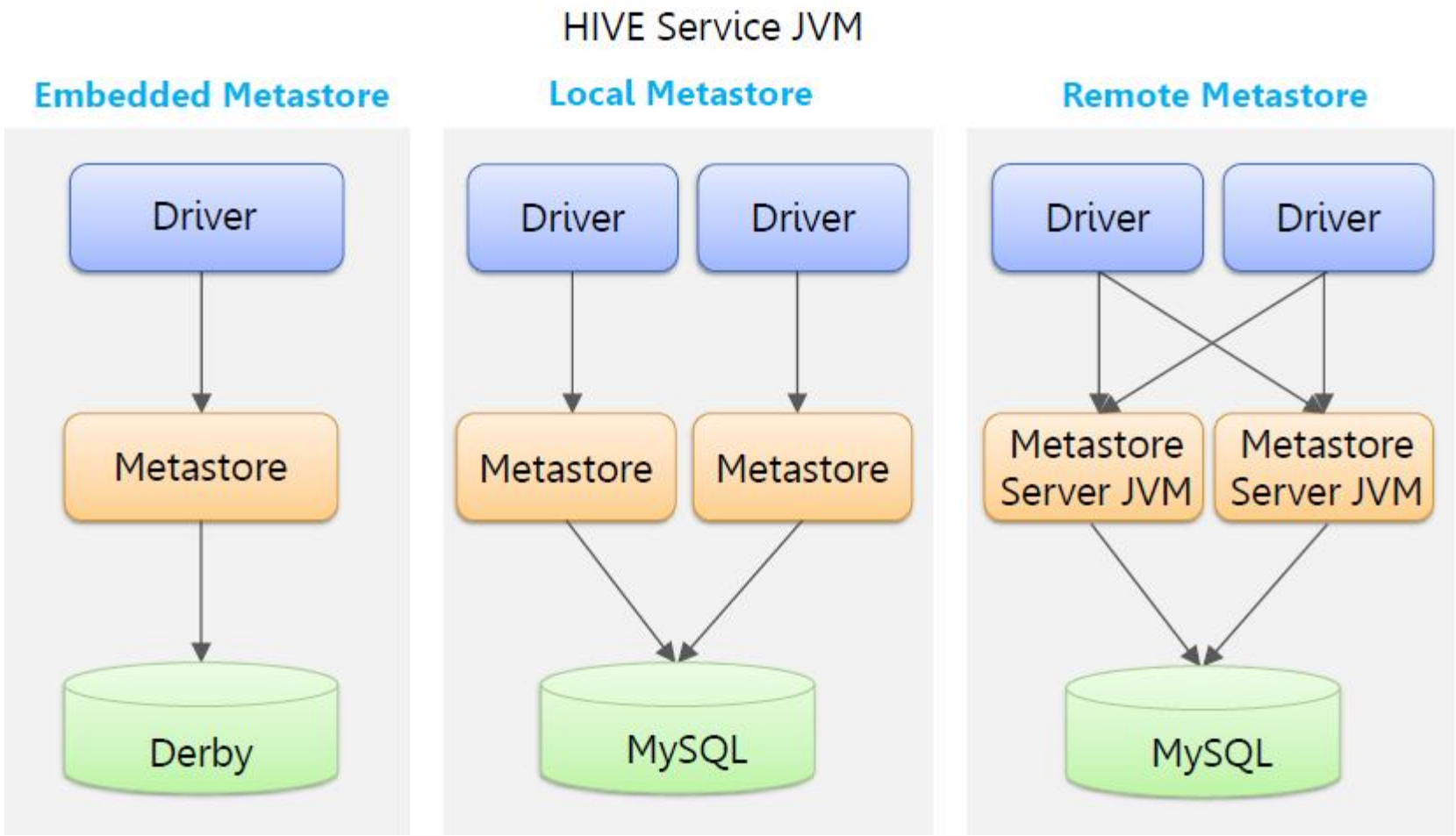
Hive Components



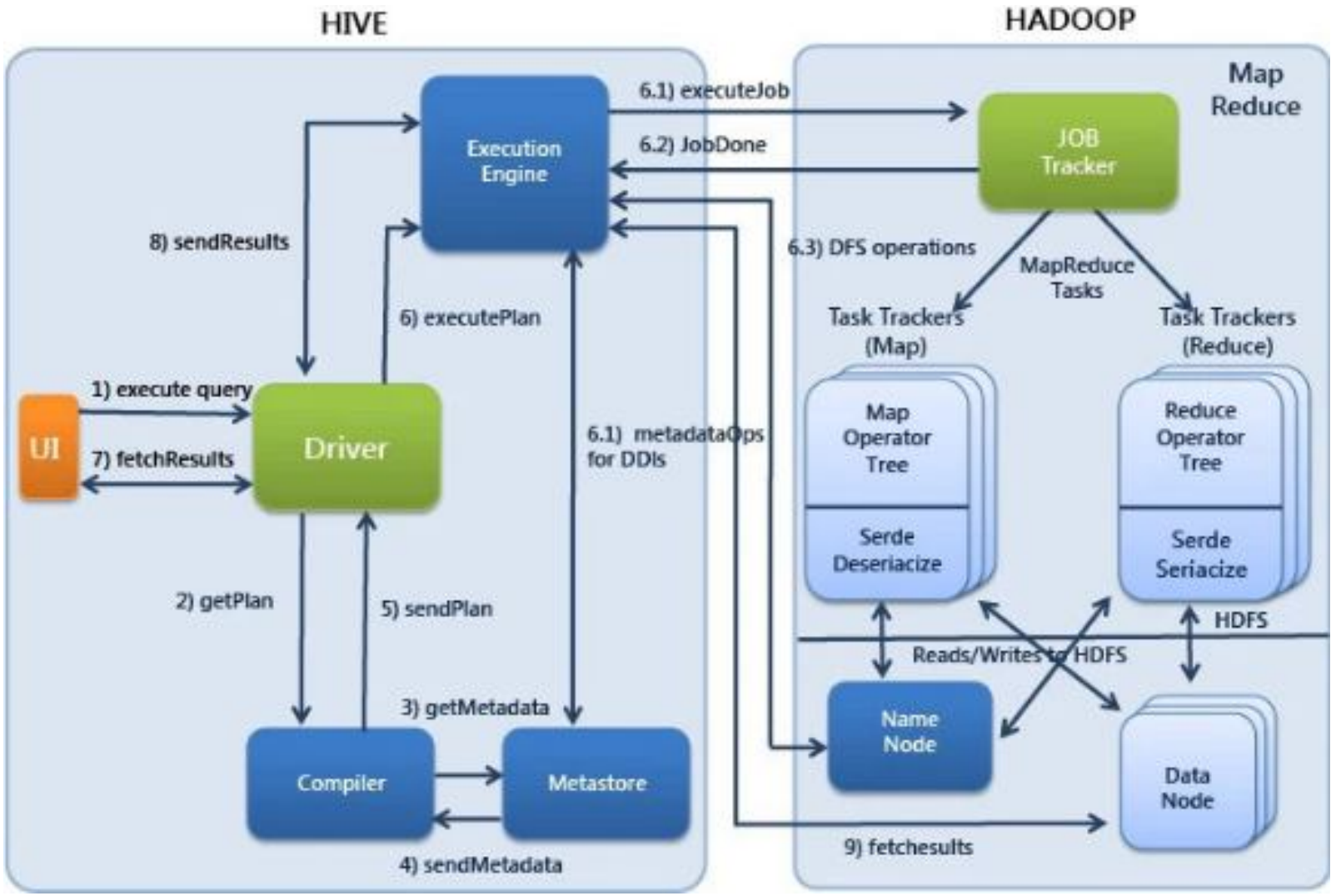
Hive Architecture



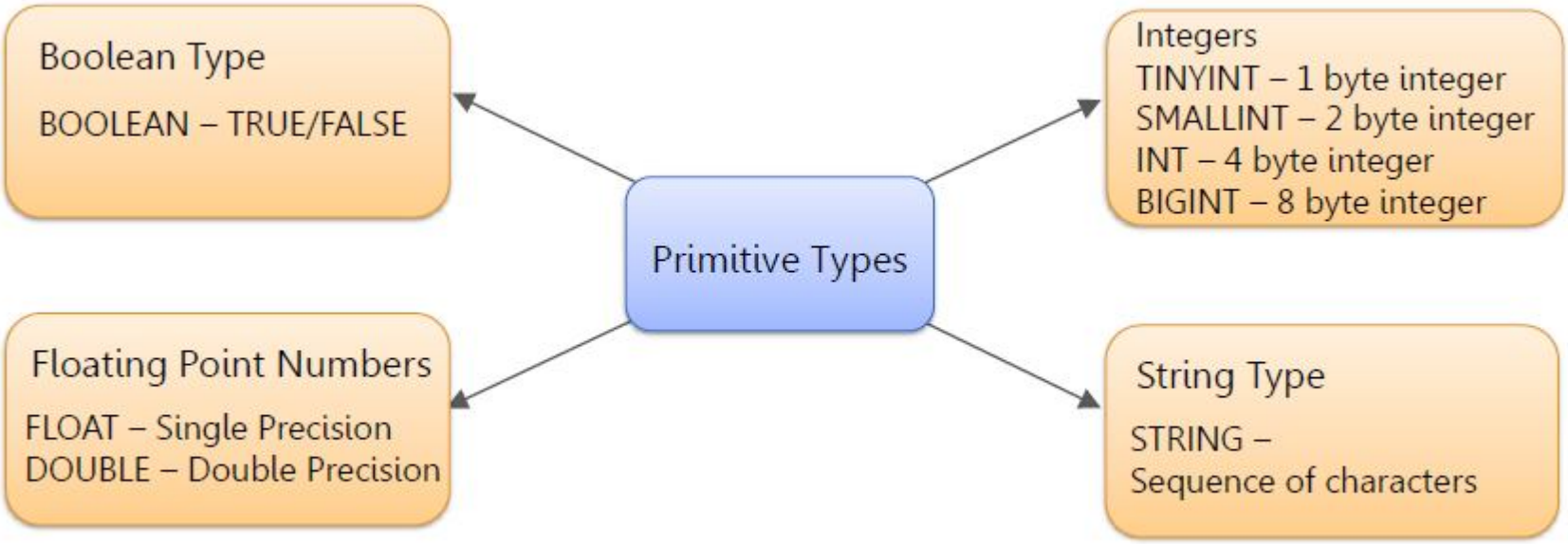
Hive Metastore



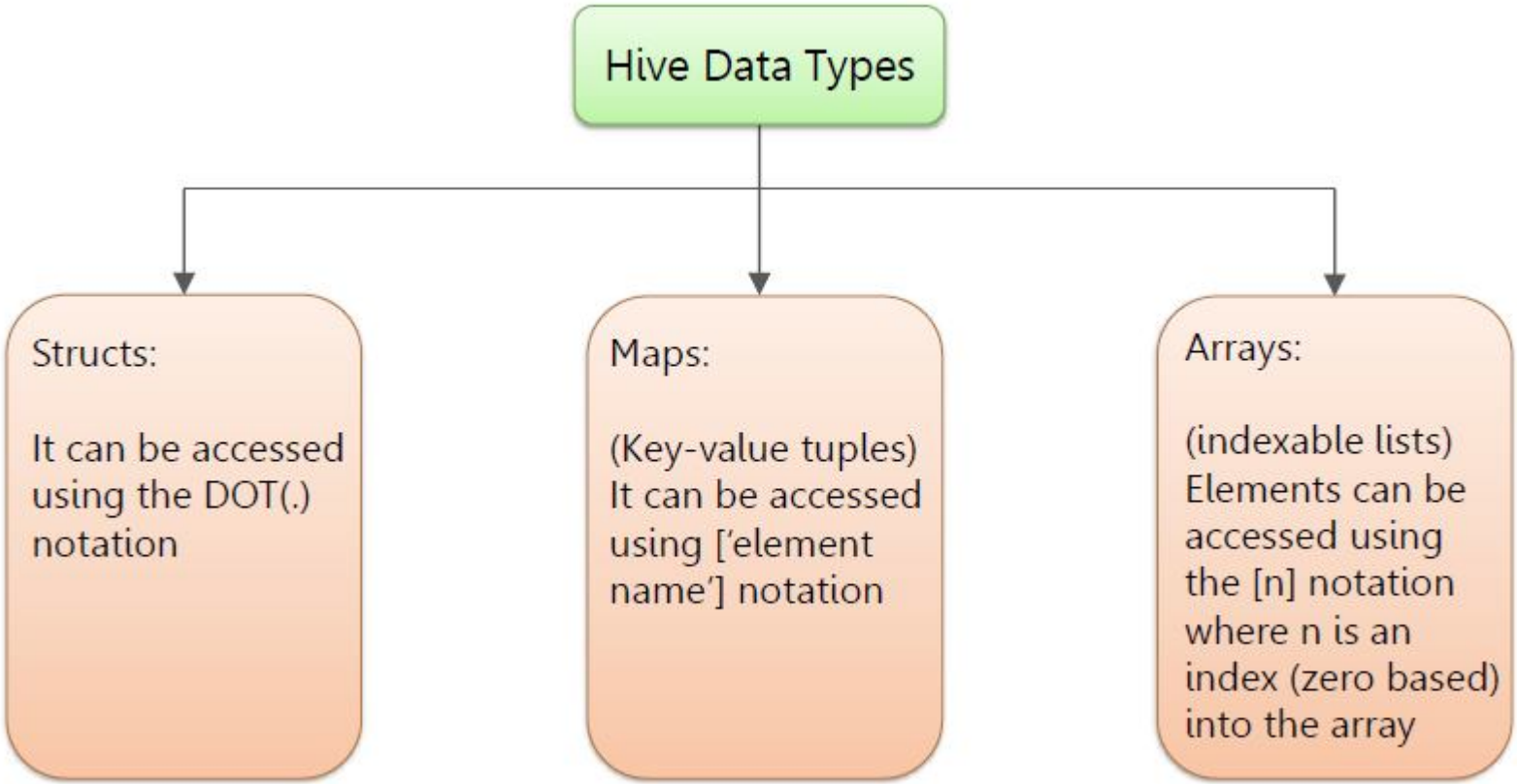
Working of Hive



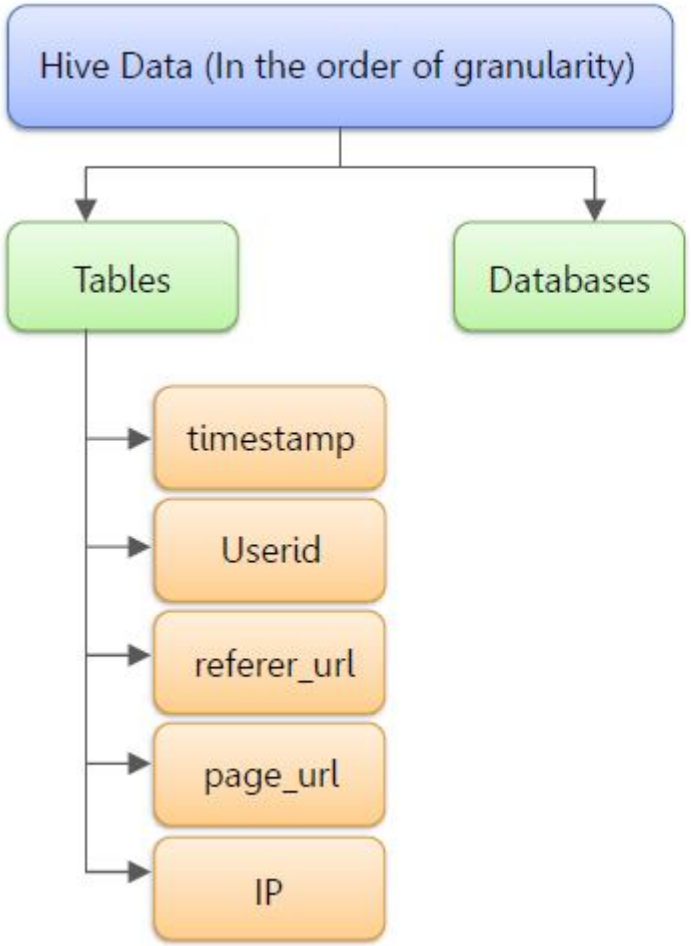
Hive Data Types



Hive Data Types (contd.)



Hadoop Data Models



Databases

- Namespaces

Tables

- Schemas in namespaces

Partitions

- How data is stored in HDFS?
- Grouping data bases on some column
- Can have one or more columns

Buckets or Clusters

- Partitions divided into buckets bases or some other column
- Use for data sampling

Hive Shell

Starting Hive CLI:

$ hive

Hive Command Line Options:

```
usage: hive
 -d,--define <key=value>          Variable substitution to apply to Hive
                                  commands. e.g. -d A=B or --define A=B
 -e <quoted-query-string>         SQL from command line
 -f <filename>                    SQL from files
 -H,--help                        Print help information
 -h <hostname>                    Connecting to Hive Server on remote host
    --hiveconf <property=value>   Use value for given property
    --hivevar <key=value>         Variable substitution to apply to hive
                                  commands. e.g. --hivevar A=B
 -i <filename>                    Initialization SQL file
 -p <port>                        Connecting to Hive Server on port number
 -S,--silent                      Silent mode in interactive shell
 -v,--verbose                     Verbose mode (echo executed SQL to the
                                  console)
```

Beeline Shell

Starting Beeline CLI:

$ *beeline*

Connect to embedded local HiveServer

beeline> *!connect jdbc:hive2:// username password*

Beeline Command Line Options:

```
-u <database url>               the JDBC URL to connect to
-c <named url>                  the named JDBC URL to connect to,
                                which should be present in beeline-site.xml
                                as the value of beeline.hs2.jdbc.url.<namedUrl>
-r                              reconnect to last saved connect url (in conjunction with !save)
-n <username>                   the username to connect as
-p <password>                   the password to connect as
-d <driver class>               the driver class to use
-i <init file>                  script file for initialization
-e <query>                      query that should be executed
-f <exec file>                  script file that should be executed
-w (or) --password-file <password file>  the password file to read password from
```

Hive Managed Tables

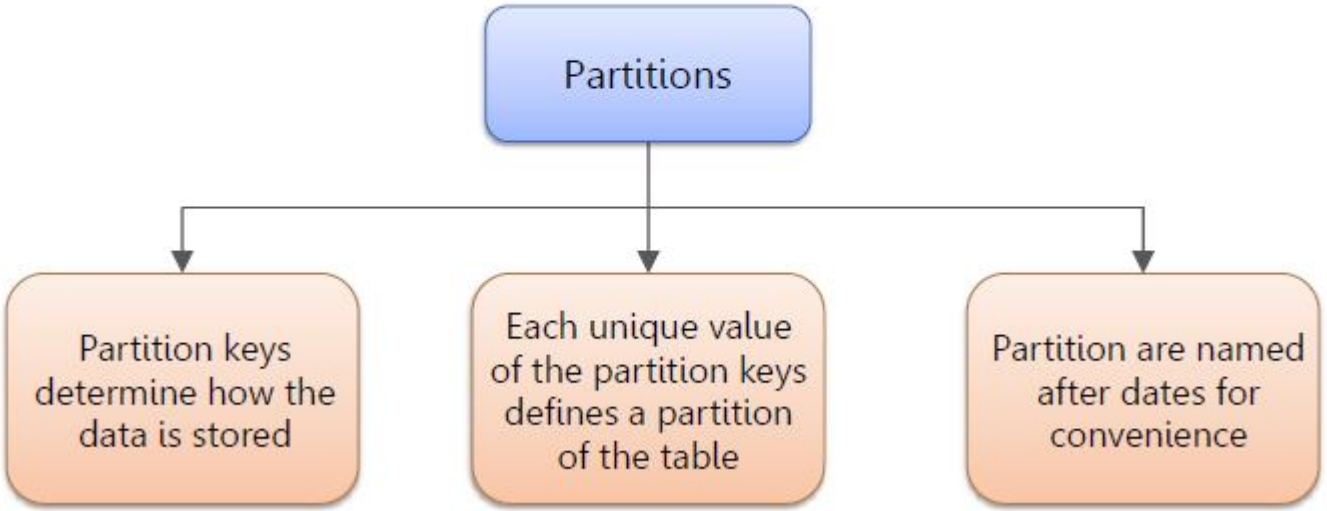
- By Default we Create the "managed" or "internal" tables in Hive
- They are called so as Hive Manages their lifecycle
- When a Managed table is dropped, Hive deletes the metadata as well as the data of the Table
- Typically used to create the user specific or group specific data

Hive External Tables

- Hive external tables are created using "external" keyword
- Hive just owns the metadata, and NOT the actual data of the external tables
- When an external table is dropped, Hive deletes ONLY the metadata of the table
- Typically used to expose the enterprise wide data as tables for different groups

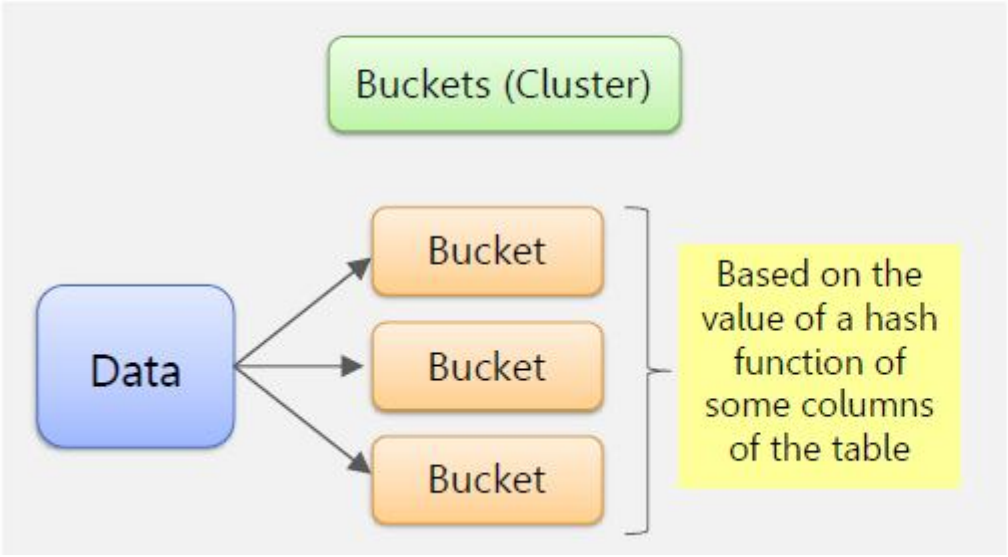
Partitioned Tables

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



Bucketed Tables

Buckets give extra structure to the data that may be used for more efficient queries



- A join of two tables that are bucketed on the same columns – including the join column can be implemented as a Map Side Join
- Bucketing by user ID means we can quickly evaluate a user based query by running it on a randomized sample of the total set of users

Hive Views

- Hive supports creation of the views on underlying tables
- A Hive view saves the query and treats it like a table
- Hive view is a logical construct and does not store any data
- Views help in simplifying the complicated select queries
- Hive executes the view query every time the view is queried
- Since Hive view queries are run every time when they are accessed, a view may fail if underlying table(s) structure change

Using custom Map Reduce with Hive

```
FROM (
FROM cv_users
SELECT TRANSFORM (cv_users.userid, cv_users.date)
USING 'map_script'
AS(dt, uid)
CLUSTER BY(dt)) map
INSERT INTO TABLE pv_users_reduced
SELECT TRANSFORM(map.dt, map.uid) USING
'reduce_script' AS (date, count);
```

Hive QL allows traditional map/reduce programmers to be able to plug in their custom mappers and reducers to do more sophisticated analysis that may not be supported by the built-in capabilities of the language

Joins in Hive

Joins

```
FROM course_view cv JOIN user edu ON (cv.userid= edu.id)
INSERT INTO TABLE cv_users
SELECT cv.*, edu.gender, edu.age
WHERE cv.date= 2011-06-06;
```

Outer Joins

```
FROM course_view cv FULL OUTER JOIN user edu ON (cv.userid=
edu.id)
INSERT INTO TABLE cv_users
SELECT cv.*, edu.gender, edu.age
WHERE cv.date= 2011-06-06;
```

UDFs in Hive

UDF Sample Code

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

Registering the class

```
CREATE FUNCTIONedu_lower AS 'com.example.edu.udf.Lower
```

Using the function

```
SELECT edu_lower(title), sum(freq) FROM titles GROUP BY edu_lower(title);
```

Thank You!