Apache Hive

hive.apache.org

جلسه چهارم

آشنایی با هایو

SQL On Hadoop

آنچه خواهیم دید

- مفاهیم پایه و معماری
- آشنایی با HiveQL
- فرمت‌های ذخیره فایل
- کارگاه عملی هایو – مقدمات
- کارگاه عملی هایو – بررسی یک مثال کاربردی

محدودیتهای هدوپ مسائل پایه

سرعت پردازش
استفاده زیاد از فایل سیستم / پردازش آفلاین

- آپاچی TEZ - یک چهارچوب اجرای تسک با محوریت DAG
- آپاچی اسپارک
- آپاچی فلینک و ....

پیچیدگی کارکرد
محدود بودن به مدل توزیع و تجمیع

- SQL : کتابخانه‌هایی که دستورات SQL را به MR تبدیل می کنند : Drill / Impala / Hive
- Script : دستورات سطح بالا را از کاربر گرفته و به MR تبدیل می کنند : Pig
- چارچوب های جایگزین : اسپارک / فلینک / ایگنایت / اپکس . ...

دسترسی تصادفی
روال معمول : اسکن خطی فایل‌ها و داده‌ها

- ساخت کتابخانه‌های واسط : Hudi /Hbase

مثالی از کدنویسی با Pig

```
A = LOAD ‘myfile’  AS (x, y, z);
B = FILTER A by x > 0;
C = GROUP B BY x;
D = FOREACH A GENERATE x, COUNT(B);
STORE D INTO ‘output’;
```

Apache Pig Releases

بعد از نه سال

- Download
- News
  - 19 June, 2017: release 0.17.0 available
  - 8 June, 2016: release 0.16.0 available
  - 6 June, 2015: release 0.15.0 available
  - 20 November, 2014: release 0.14.0 available
  - 4 July, 2014: release 0.13.0 available

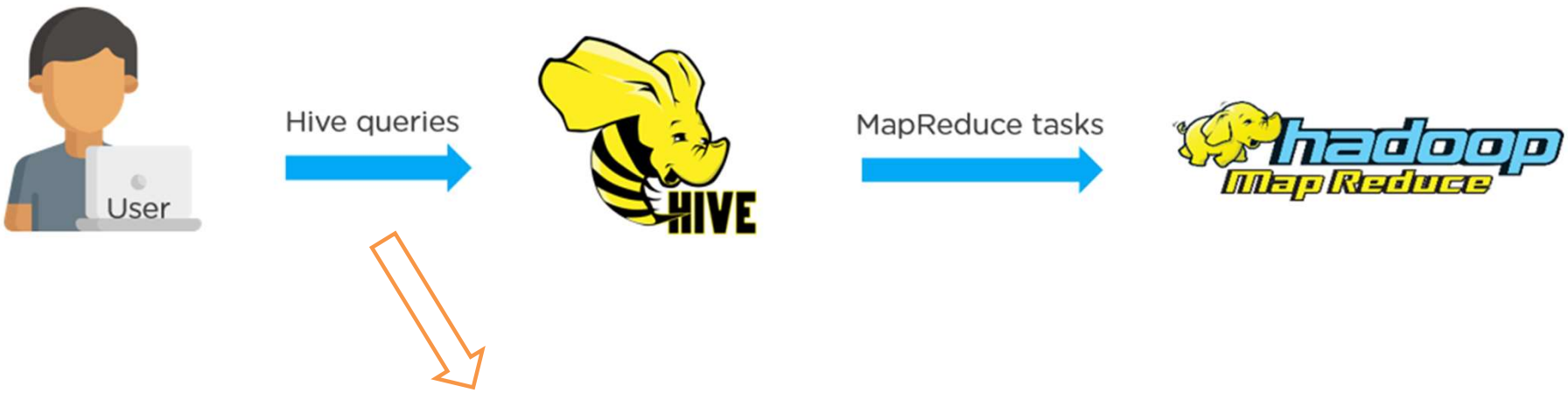
تعریف رسمی بنیاد آپاچی از هایو

The Apache Hive ™ :

- **data warehouse** software
- facilitates reading, writing, and managing **large datasets**
- residing in **distributed storage**
- using **SQL**

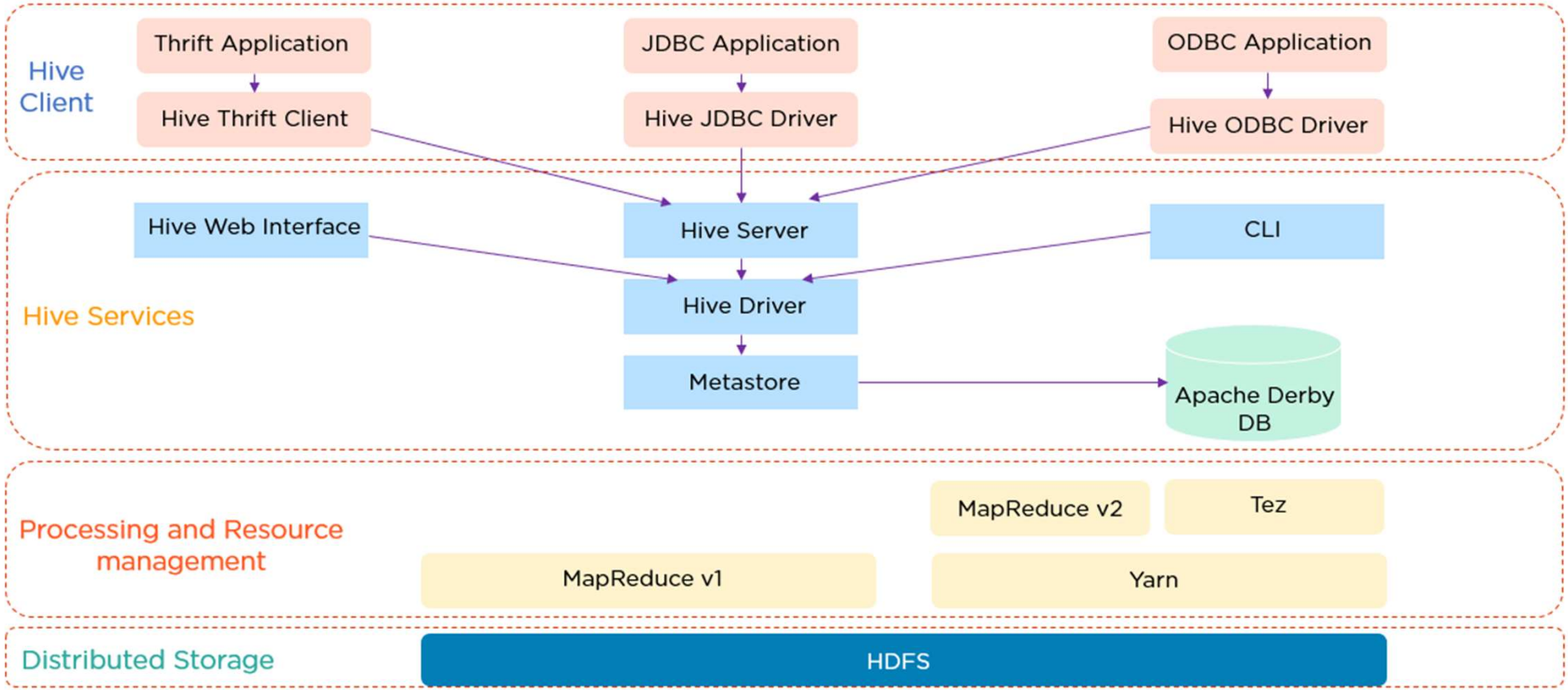
انبار داده یا Data Warehouse پایگاه داده‌ای است که برای گزارش‌گیری و تحلیل داده به کار می‌رود و بعنوان هسته اصلی یک سیستم هوش تجاری در سامانه‌های داده‌محور امروزی به شمار می‌آید

هایو در یک نگاه

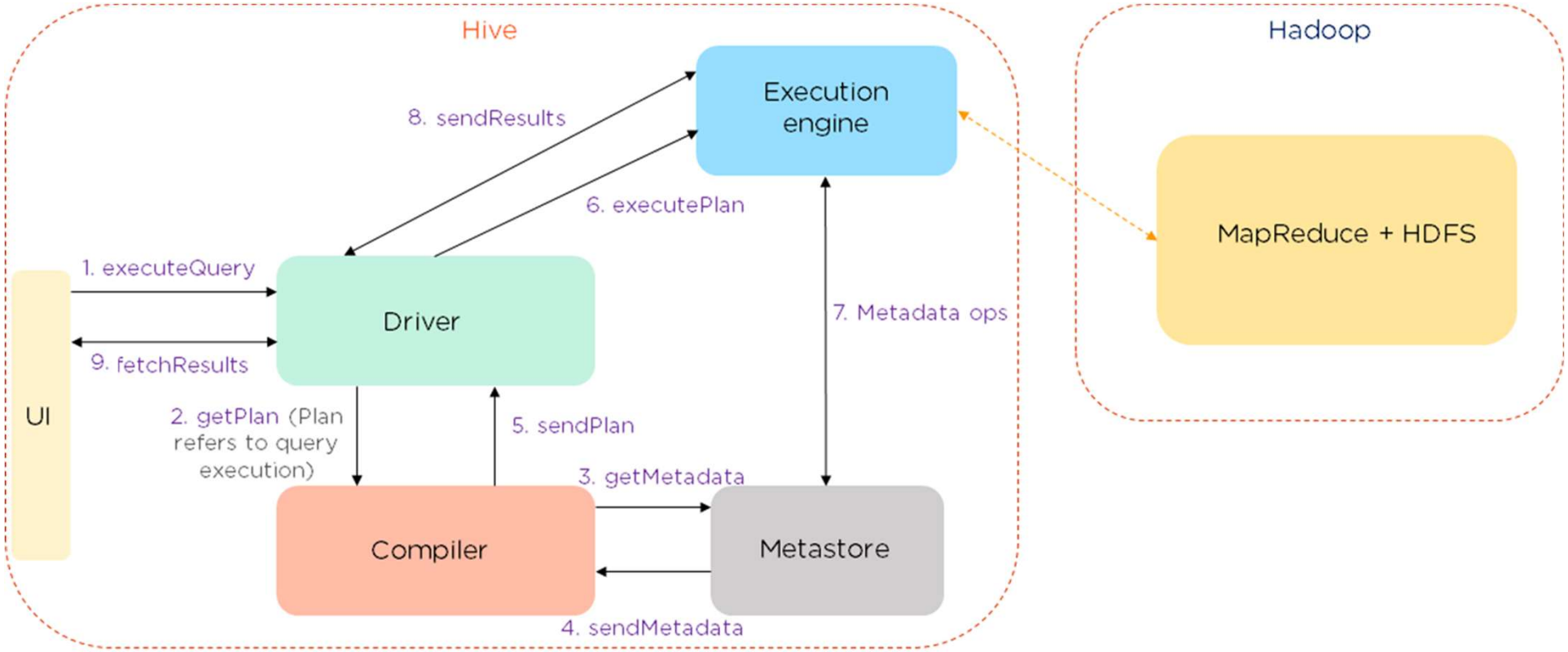


HiveQL

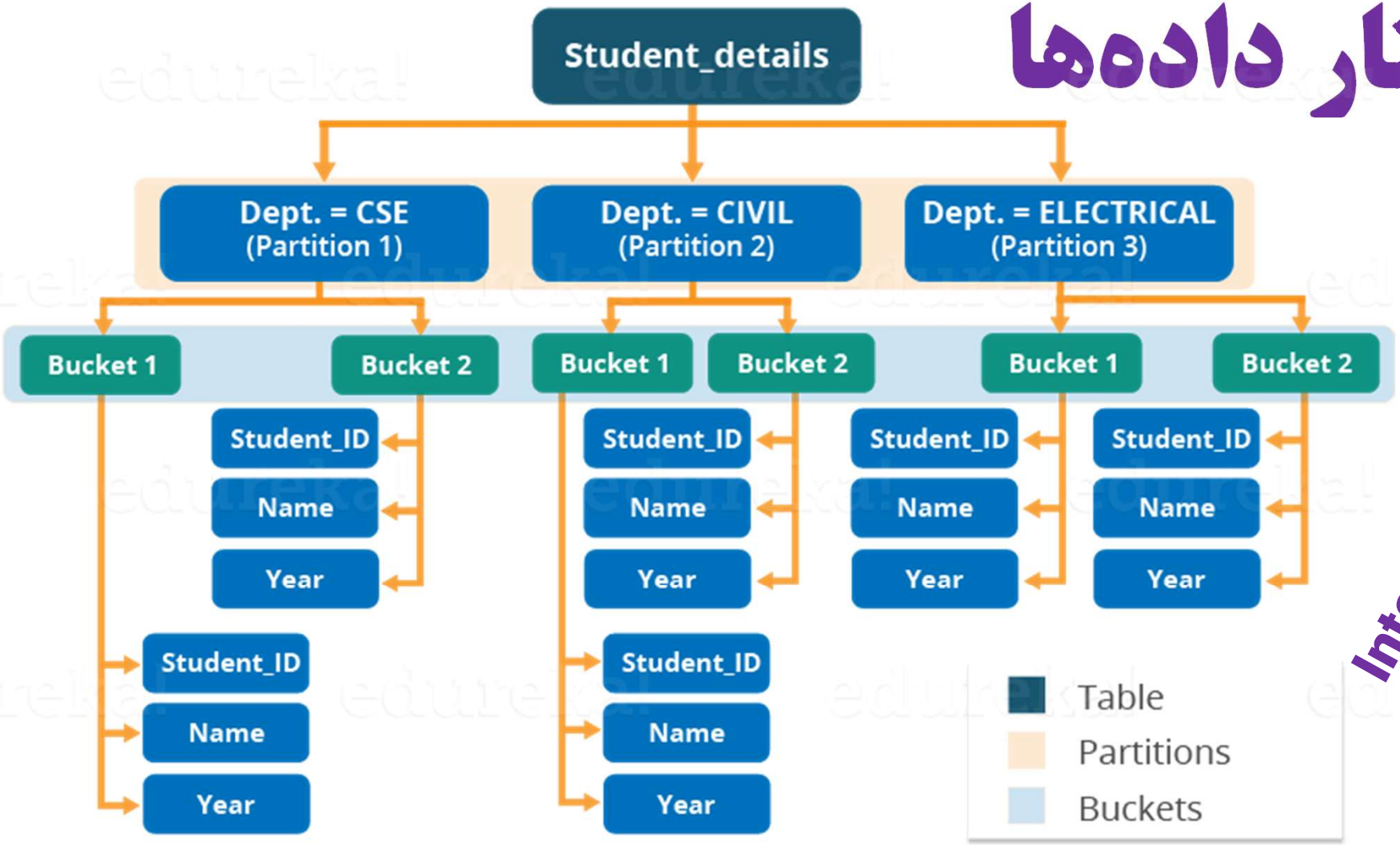
معماری هایو


Hive Client
Thrift Application
Hive Thrift Client
JDBC Application
Hive JDBC Driver
ODBC Application
Hive ODBC Driver
Hive Services
Hive Web Interface
Hive Server
CLI
Hive Driver
Metastore
Apache Derby DB
Processing and Resource management
MapReduce v2
Tez
MapReduce v1
Yarn
Distributed Storage
HDFS

جریان کار در هایو



ساختار داده‌ها



Internal Tables

ایجاد جدول و طراحی ساختار

```
CREATE TABLE table_name ...
PARTITIONED BY
 (partition۱ data_type, partition۲ data_type,....)
CLUSTERED BY
 (column_name۱, column_name۲, ...)
SORTED BY (column_name [ASC|DESC], ...)]
 INTO num_buckets BUCKETS;
```

Internal Tables

محدودیت‌های هایو

- هایو یک دیتابیس رابطه‌ای نیست
- امکان حذف یا به روزرسانی داده‌ها را ندارد
  ← رونویسی یا افزودن
- امکانات پیشرفته SQL، بسیار محدود پشتیبانی می شوند

HiveQL — زبان کوئری هایو

- **Philosophy**
  SQL like constructs + Hadoop Streaming

- **Query Operators in initial version**
  Projections
  Joins
  Group by
  Sampling

- **Output of these operators can be:**
  passed to Streaming mappers/reducers
  can be stored in another Hive Table
  can be output to HDFS files
  can be output to local files

انواع داده‌های پایه

۱.Numeric Data types - Data types like integral, float, decimal
۲.String Data type - Data types like char, string
۳.Date/ Time Data type - Data types like timestamp, date, interval
۴.Miscellaneous Data type - Data types like Boolean and binary

انواع داده‌های پیشرفته

۱. **Arrays** - A collection of the same entities. The syntax is: array<data_type>
۲. **Maps** - A collection of key-value pairs and the syntax is map<primitive_type, data_type>
۳. **Structs** - A collection of complex data with comments. Syntax: struct<col_name : data_type [COMMENT col_comment],.....>

مثالی از ایجاد جدول

```
CREATE EXTERNAL TABLE page_view(
     viewTime INT, userid BIGINT,
      page_url STRING, referrer_url STRING,
      ip STRING COMMENT 'IP Address of the User',
      country STRING COMMENT 'country of origination‘
)
 COMMENT 'This is the staging page view table'
 ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ‘,'
 STORED AS TEXTFILE
 LOCATION ‘/data/page_views';
```

مثالی از ایجاد جدول با پارتیشن

```
CREATE TABLE page_view(
      viewTime INT, userid BIGINT,
       page_url STRING, referrer_url STRING,
       ip STRING COMMENT 'IP Address of the User',
       country STRING COMMENT 'country of origination‘
)
 COMMENT 'This is the staging page view table‘
 PARTITIONED BY(dt STRING, country STRING)
  CLUSTERED BY (userid)
SORTED BY (viewTime) INTO 32 BUCKETS
 ROW FORMAT DELIMITED
   FIELDS TERMINATED BY '\001'
   COLLECTION ITEMS TERMINATED BY '\002'
   MAP KEYS TERMINATED BY '\003'
 STORED AS SEQUENCEFILE
```

انواع گزینه های مرتب سازی

ORDER BY x: guarantees global ordering, but does this by pushing all data through just one reducer.

SORT BY x: orders data at each of N reducers, but each reducer can receive overlapping ranges of data. You end up with N or more sorted files with overlapping ranges.

DISTRIBUTE BY x: ensures each of N reducers gets non-overlapping ranges of x, but doesn't sort the output of each reducer.

CLUSTER BY x: ensures each of N reducers gets non-overlapping ranges, then sorts by those ranges at the reducers.

مثالی از ایجاد جدول با ساختار پیچیده

```
CREATE TABLE struct_demo
(
  id BIGINT,
  name STRING,
 employee_info STRUCT < employer: STRING, id: BIGINT, address: STRING >,
 places_lived ARRAY <STRUCT <street: STRING, city:STRING,country:STRING >>,
  memorable_moments MAP < STRING, STRUCT < year: INT, place: STRING,
details: STRING >>,
  current_address STRUCT < street_address: STRUCT <street_number:
INT, street_name: STRING, street_type: STRING>, country: STRING,
postal_code: STRING >
)
STORED AS PARQUET;
```

نحوه جستجوی اطلاعات پیشرفته

```
SELECT id, employee_info.id FROM struct_demo;

SELECT id, name, street, city, country
  FROM struct_demo, struct_demo.places_lived;

SELECT id, name, mm.key, mm.value.year,
mm.value.place, mm.value.details
  FROM struct_demo, struct_demo.memorable_moments AS
mm
WHERE mm.key IN ('Birthday','Anniversary','Graduation');
```

بارگذاری داده‌ها از فایل به جدول

```
LOAD DATA LOCAL INPATH
'usr/data/std_details.txt'
OVERWRITE INTO TABLE
std_details;
```

چند کوئری پیشرفته در هایو

```
INSERT OVERWRITE TABLE actions_users
SELECT u.id, actions.date
    FROM (
    SELECT av.uid AS uid, date
    FROM action_video av
    WHERE av.date = '2008-06-03'
    UNION ALL
    SELECT ac.uid AS uid, date
    FROM action_comment ac
    WHERE ac.date = '2008-06-03'
) actions JOIN users u ON (u.id = actions.uid);
```

چند کوئری پیشرفته در هایو

```
FROM pv_users
  INSERT INTO TABLE pv_gender_uu
  SELECT pv_users.gender, count(DISTINCT
  pv_users.userid)
  GROUP BY(pv_users.gender)
  INSERT INTO TABLE pv_ip_uu
  SELECT pv_users.ip, count(DISTINCT pv_users.id)
  GROUP BY(pv_users.ip);
```

چند کوئری پیشرفته در هایو

```
FROM pv_users
INSERT INTO TABLE pv_gender_sum
   SELECT pv_users.gender,
   count_distinct(pv_users.userid)
   GROUP BY(pv_users.gender)
INSERT INTO DIRECTORY '/user/tmp/pv_age_sum.dir'
   SELECT pv_users.age, count_distinct(pv_users.userid)
   GROUP BY(pv_users.age)
INSERT INTO LOCAL DIRECTORY '/home/me/pv_age_sum.dir'
      FIELDS TERMINATED BY ',' LINES TERMINATED BY \013
   SELECT pv_users.age, count_distinct(pv_users.userid)
   GROUP BY(pv_users.age);
```

چند کوئری پیشرفته در هایو

```
EXPLAIN EXTENDED SELECT empid, deptname
FROM emps
JOIN depts
ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2017-01-01'
AND hire_date <= '2019-01-01';
```

```
CREATE MATERIALIZED VIEW mv1
AS
SELECT dest, origin, count(*)
FROM flights_hdfs
GROUP BY dest, origin;
```

چند کوئری پیشرفته در هایو

```
create external table logs (
 host string,req_time string,req_code string,
 req_url string,rep_code string,rep_bytes string
)
row format serde
’org.apache.hadoop.hive.contrib.serde2.RegexSerD
e’
with serdeproperties (
"input.regex" = ’([^ ]*) - - \\[(.*)\\] "([^ ]*)
([^ ]*).*" ([^ ]*) ([^ ]*)’
)
stored as textfile
location "/data/logs";
```

چند کوئری پیشرفته در هایو

In movie table :genres

genres : drama | action | fantasy

```
from (select explode (genres) as genre from movies)
genres
select genres.genre, count(*) as popularity
group by genres.genre
order by popularity desc;
```

ایندکسینگ در هایو

Table

key	string
value	string

Index_Table

key	string
_bucketname	string
_offsets	array<bigint>

```
 CREATE INDEX inedx_salary ON TABLE
employee(salary)
AS
'org.apache.hadoop.hive.ql.index.compact.Compact
IndexHandler';
```

دستورات تعریف، نمایش و تغییر ساختار داده

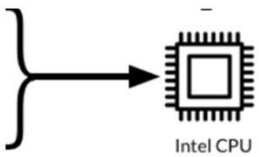
DDL Command	Function
CREATE	It is used to create a table or Database
SHOW	It is used to show Database, Table, Properties, etc
ALTER	It is used to make changes to the existing table
DESCRIBE	It describes the table columns
TRUNCATE	Used to permanently truncate and delete the rows of table
DELETE	Deletes the table data, but, can be restored

قالب ذخیره داده‌ها : سطری / ستونی

	day	location	product	sale
row 1	2017-01-01	l1	p1	300
row 2	2017-01-01	l1	p2	40
row 3	2017-01-01	l2	p1	44
row 4	2017-02-01	l1	p1	200

Traditional Memory Buffer	
row 1	2017-01-01
	l1
	p1
	300
row 2	2017-01-01
	l1
	p2
	40
row 3	2017-01-01
	l2
	p1
	44

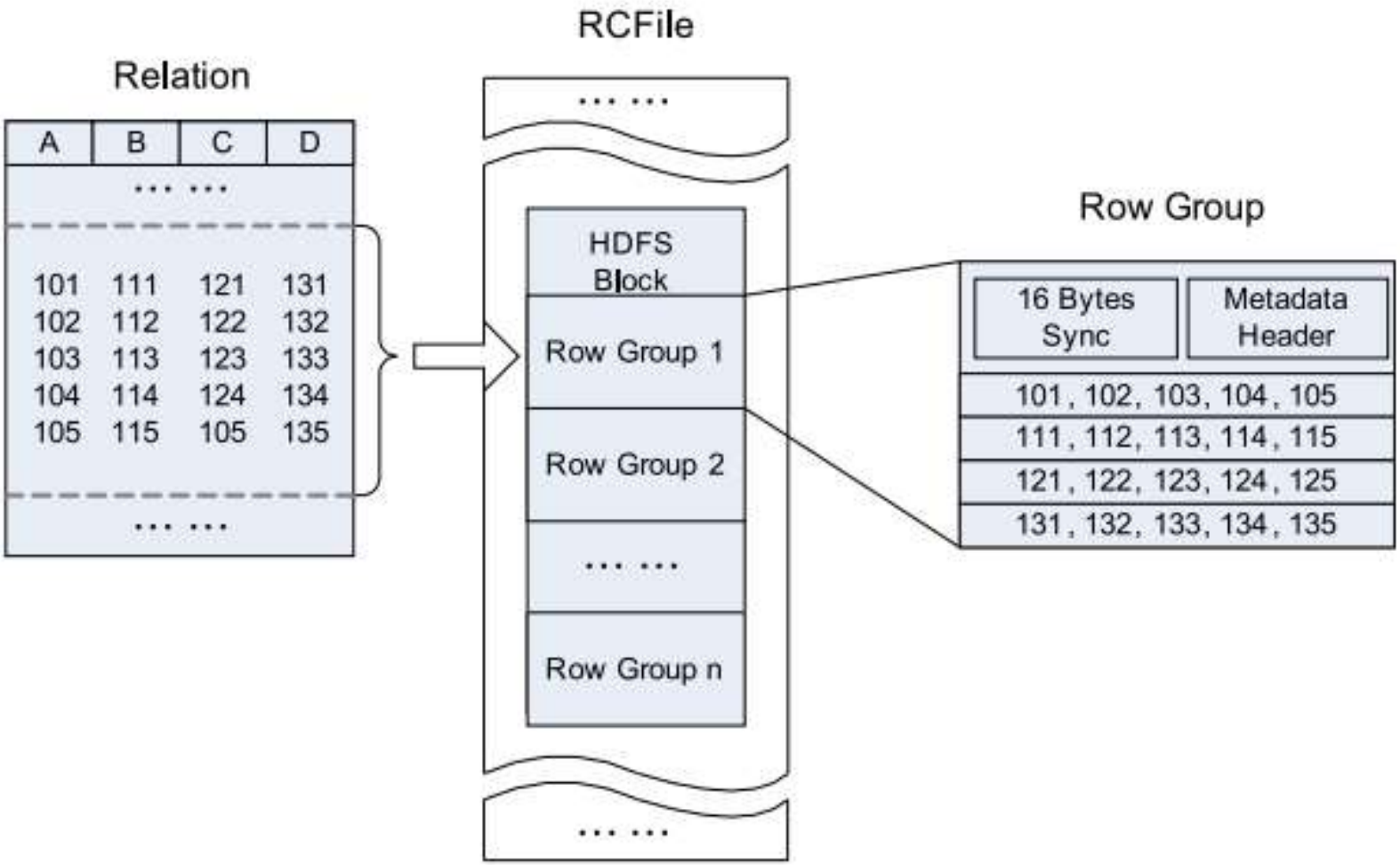
Columnar Storage	
day	2017-01-01
	2017-01-01
	2017-01-01
	2017-01-02
location	l1
	l1
	l2
	l1
product	p1
	p2
	p1
	p1



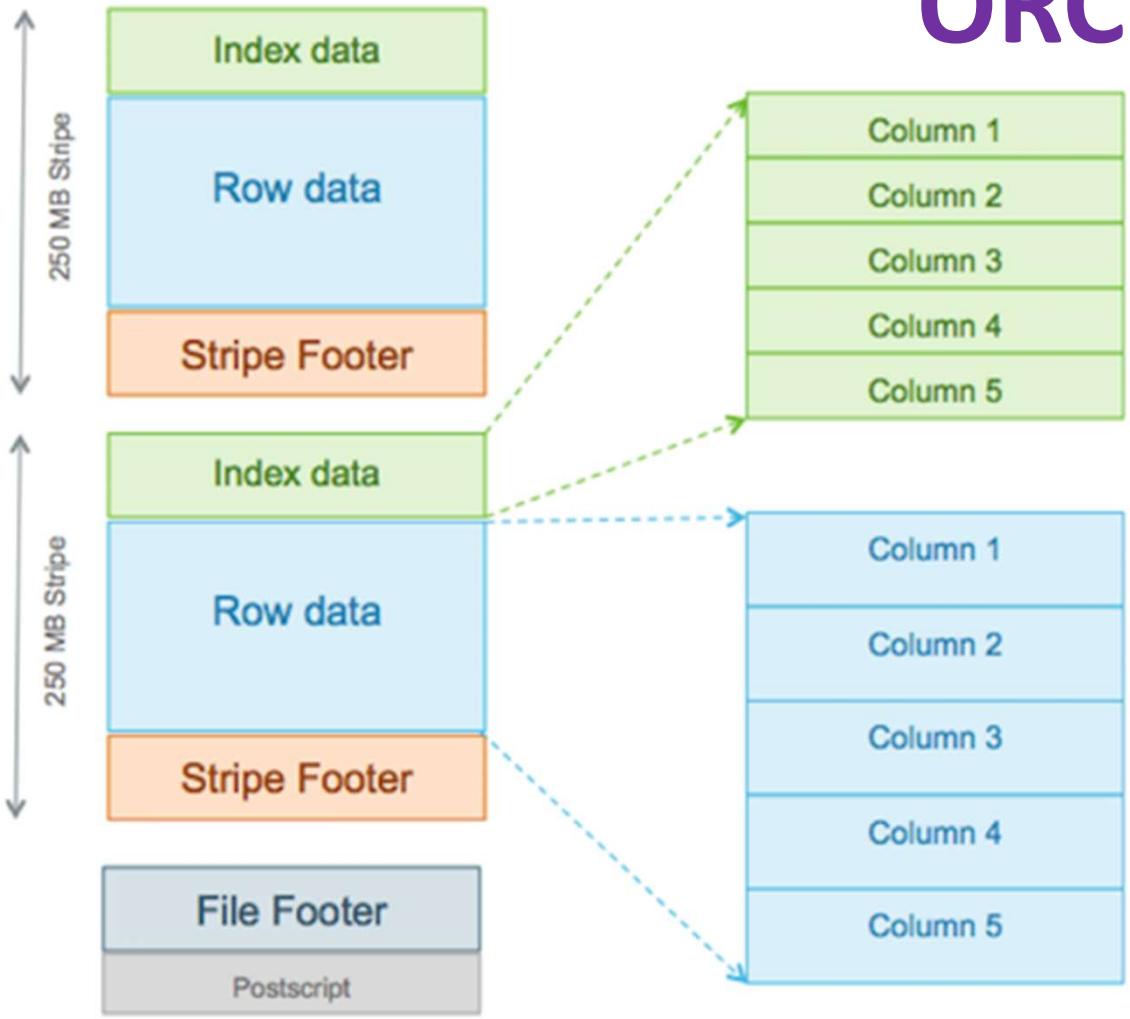
- Text File
- Sequence
- RC File
- ORC
- PARQUET
- AVRO

قالب ذخیره فایل در هایو

ساختار فایل RC



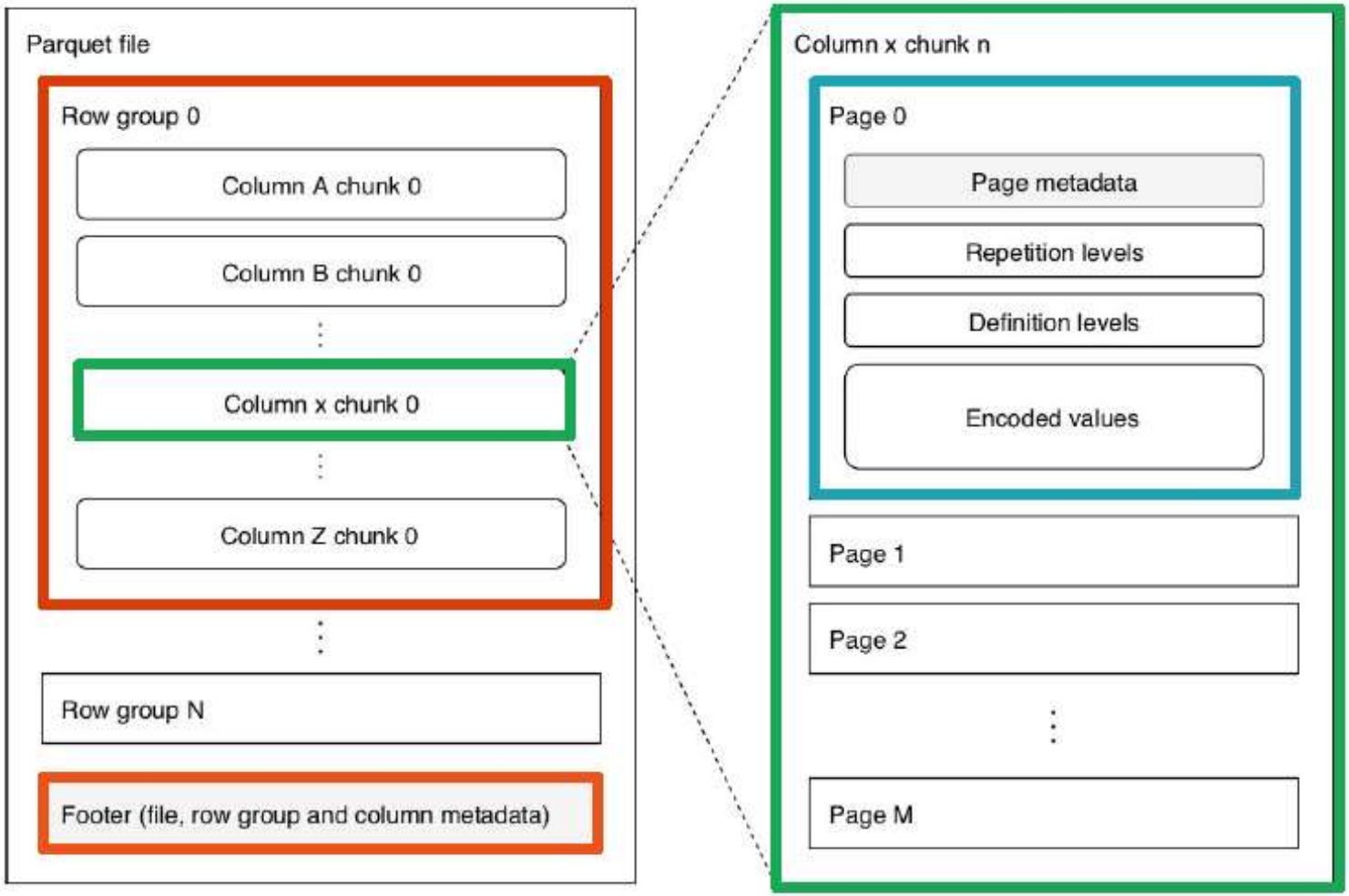
ساختار فایل ORC



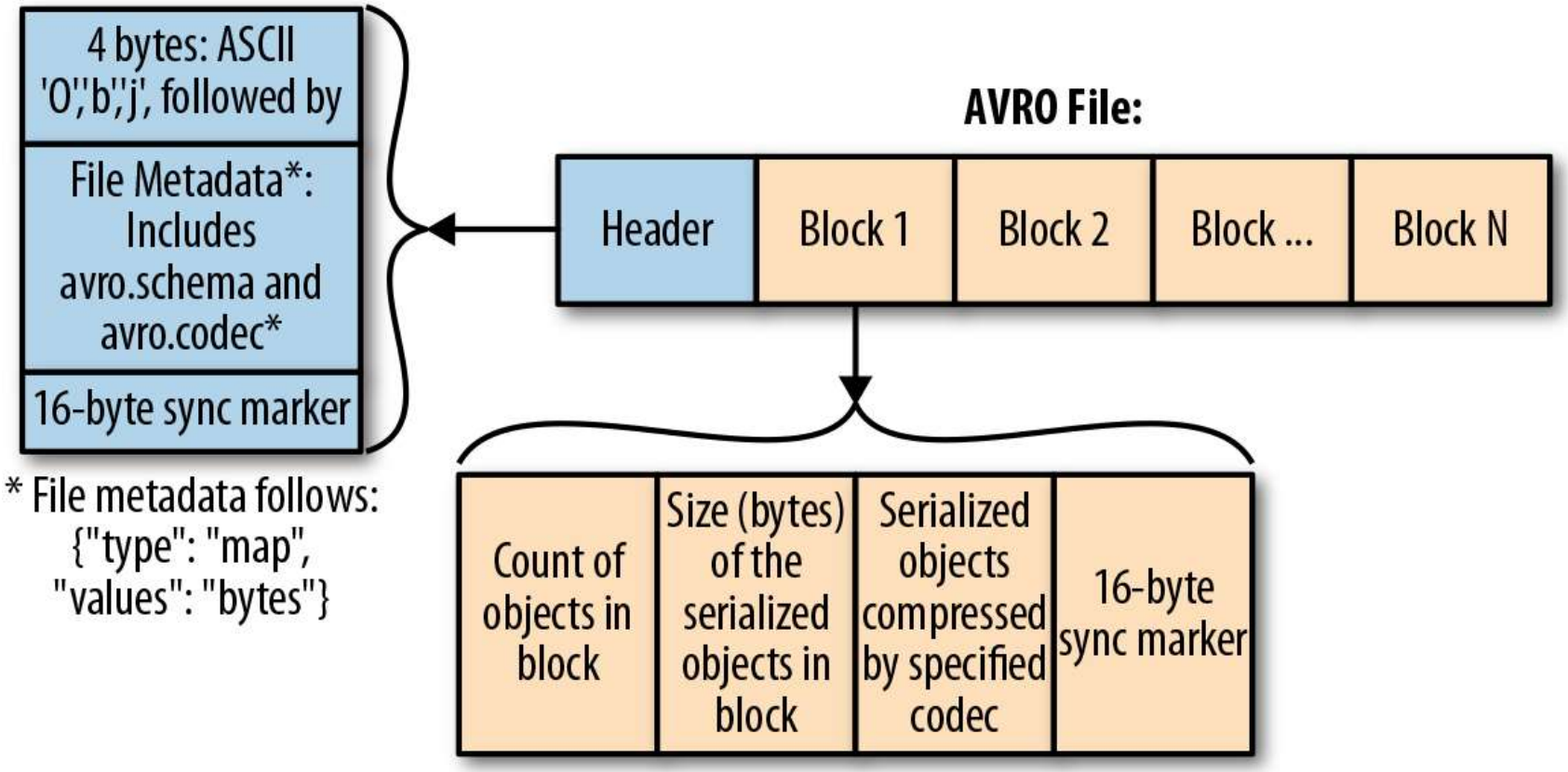

```
CREATE TABLE a_orc STORED AS
ORC AS SELECT * FROM A;

STORED AS orc tblproperties
("orc.compress"="Zlib");
```

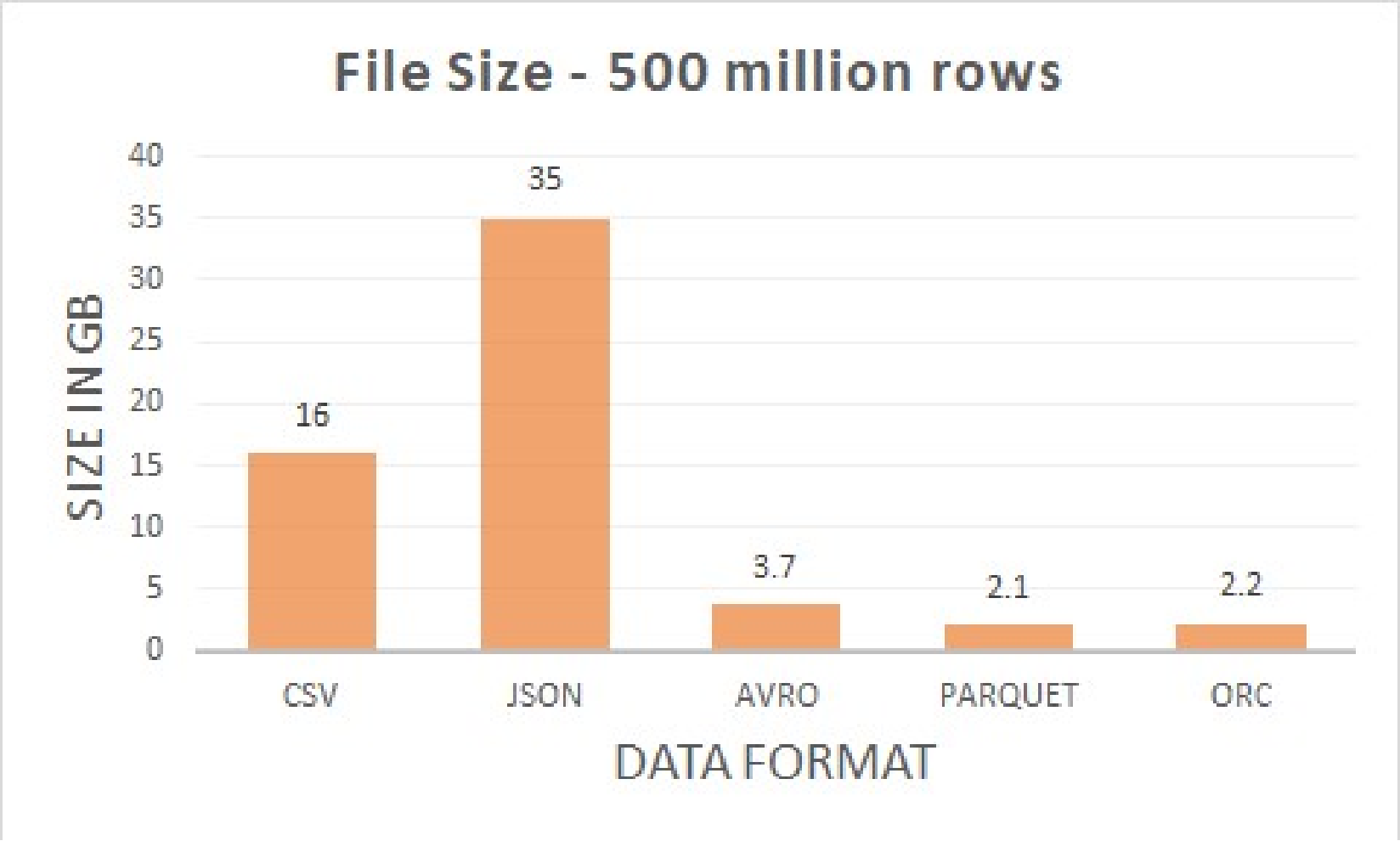
ساختار فایل پارکت



ساختار فایل AVRO



مقایسه فضای ذخیره سازی

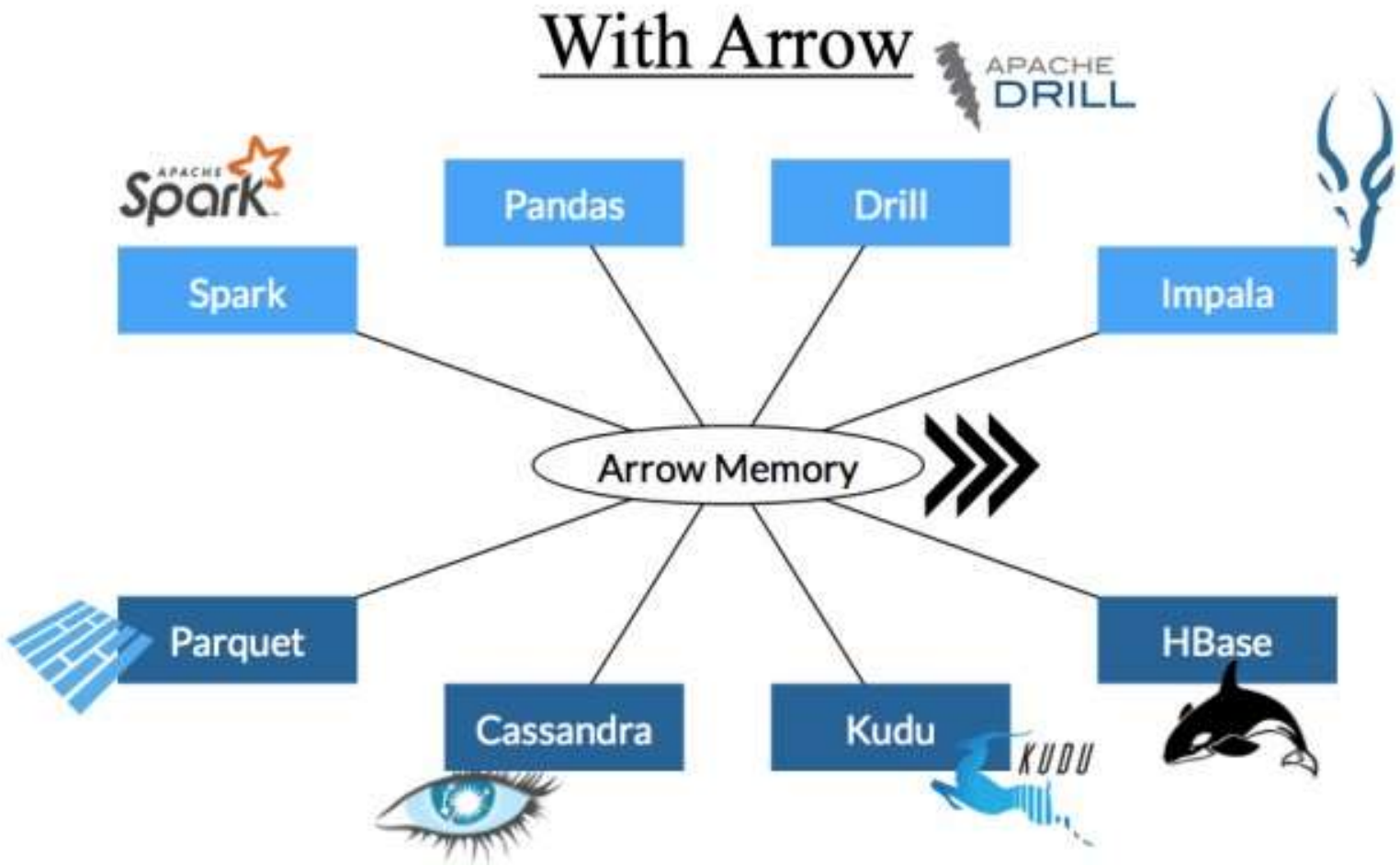


BIG DATA FORMATS COMPARISON

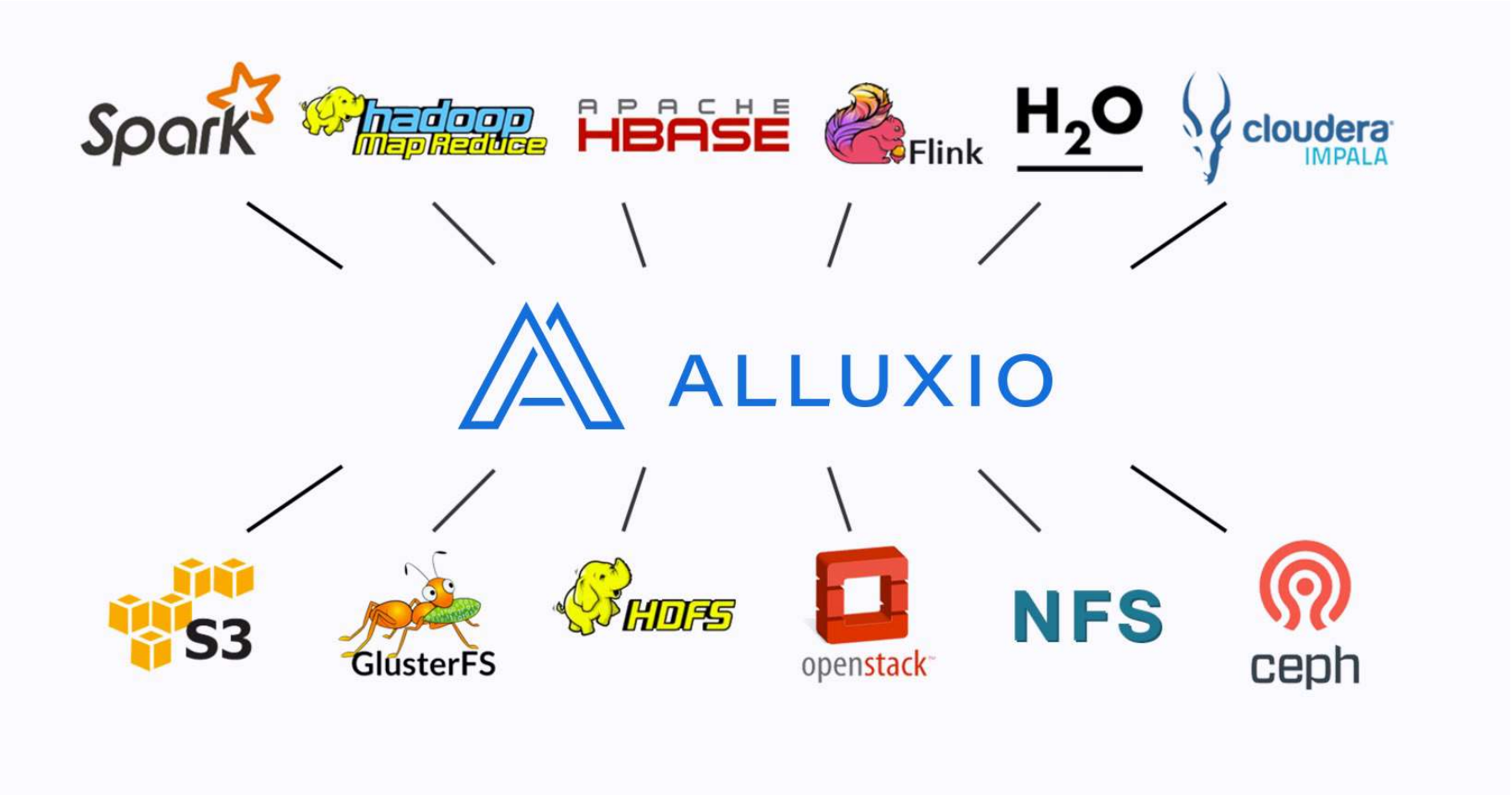
	Avro	Parquet	ORC
Schema Evolution Support			
Compression			
Splitability			
Most Compatible Platforms	Kafka, Druid	Impala, Arrow Drill, Spark	Hive, Presto
Row or Column	Row	Column	Column
Read or Write	Write	Read	Read

Source: Nexla analysis, April 2018

Apache Arrow



Alluxio


Spark
hadoop MapReduce
APACHE HBASE
Flink
H2O
cloudera IMPALA
ALLUXIO
S3
GlusterFS
HDFS
openstack
NFS
ceph

منابع و مطالعات بیشتر