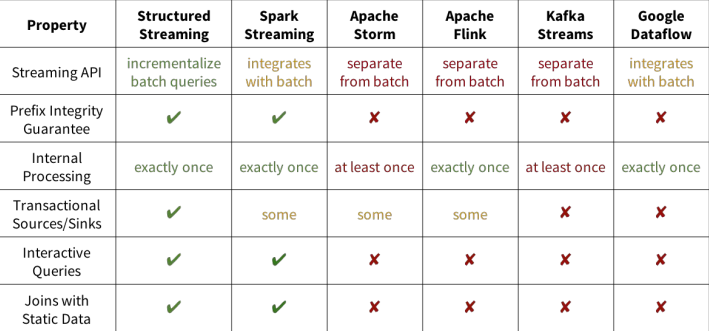
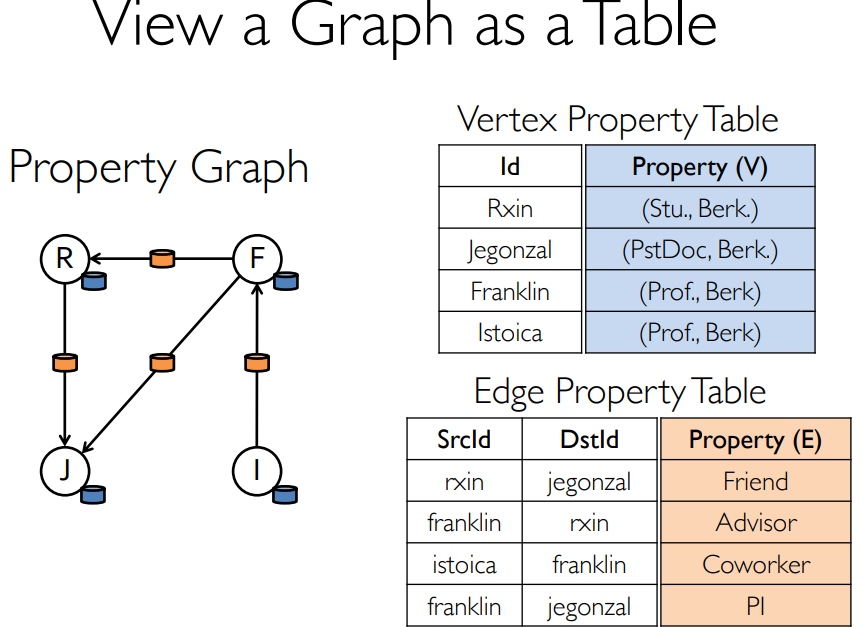
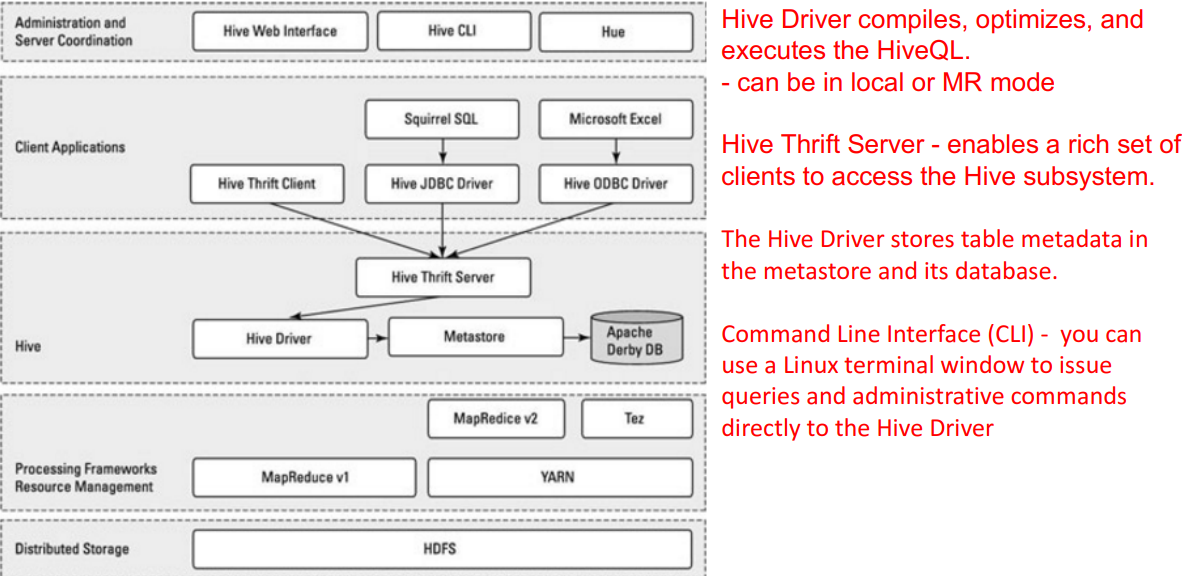
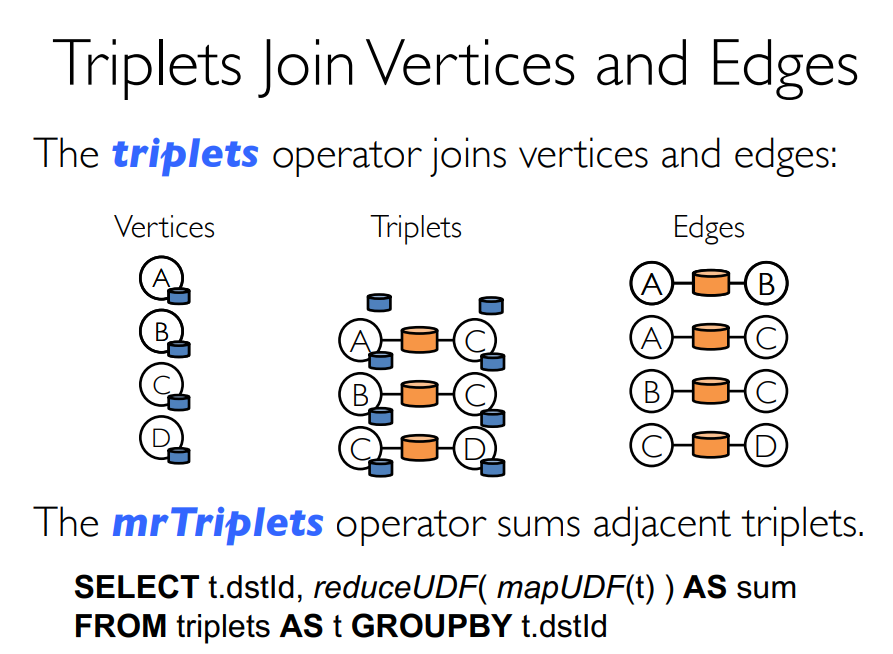
Tables and Graphs are composable views of the same physical data





A vertex is part of a triangle when it has two adjacent vertices with an edge between them.

Data units in order of granularity in Hive:

• Databases: Namespaces that separate tables and other data units from naming confliction.

• Tables: Homogeneous units of data which have the same schema.

• Partitions: Each Table can have one or more partition Keys which determines how the data is stored

• Buckets (or Clusters): Data in each partition may in turn be divided into Buckets based on the value of a hash function of some column of the Table

Impala

• Cloudera'a Massively Parallel Processing (MPP) SQL query engine.

• Impala brings scalable parallel database technology to Hadoop,

enabling low **latency and high concurrency** SQL queries.

- This is something not possible with Hive, HBase, etc

• Impala is promoted for data scientists to perform analytics on data

stored in Hadoop via SQL or other BI tools.

• You can get interactive queries in (near) real-time.

• It consists of different daemon processesthat run on specific hosts

within your cluster.

Impala implements a distributed architecture based on daemon

processesthat are responsible for all aspects of query execution and

that run on the same machines as the rest of the Hadoop

infrastructure

• Impala is the highest performing SQL-on-Hadoop system, especially

under multi-user workloads [see attached paper for details

Impala>sparksql>Presto>Hive

CAP

AP:Cassandra, simpleDB

CP:mongo, scalaries, HBase

CA: RDBM(mysql)

**Mongo**

• Based on documents \*think of it as a record\*

• A record in MongoDB is a document, which is

a data structure composed of field and value

pairs

• Documents are stored in collections \*think of

it as tables\*

• Collections do not require schema

specification – dynamic schema.

MongoDB stores data in the form of

documents, which are JSON-like field and

value pairs

Mongo DB, nosql

individual record -> document

group of record -> collecction

group of collection -> database

use <Databasename>;

db.<colletion\_name>.commandName

db.restaurants.findOne()

db.res.find({cuisine: "Italian", borough: "Manhattan"}).count()

db.orders.mapReduce(

function() { emit(this.cust\_id, this.amount); },

function(key, values) { return Array.sum(values)},

{ query: {status: "A"},

out: "order\_totals"

}

).find()

db.restaurants.mapReduce(

function() { emit(this.cuisine, 1); },

function(key, values) { return Array.sum(values)},

{ query: {borough: "Manhattan"},

out: "cusine\_totals"

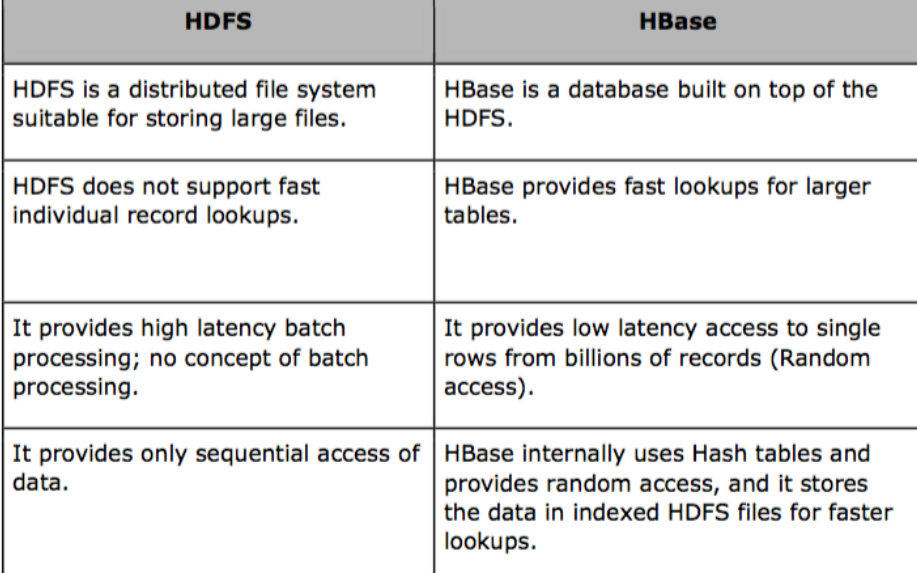
}

).find()

selection db.res.find({condition})

projection db.res.find({},{name: 1, cuisine:1, \_id:0, borough:0})

**Hbase**

HBase is a distributed column-oriented database built on top of the Hadoop file system. 

• It is an open-source project and is horizontally scalable.

HBase vs RDBMS

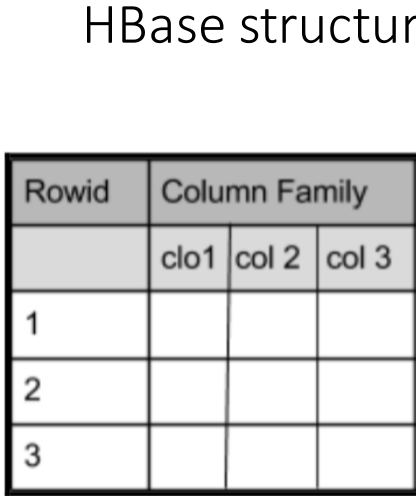
• HBase is schema-less, doesn't have ACID properties

RDBMS has schema and follows ACID properties

• HBase is non-transactional

RDBMS is transactional

• HBase is good for semi-structured and structured data

RDBMS is for structured data

• HBase made up of an HBase master node orchestrating a cluster of one or

more regionserver workers

• Master's roles: - boostrapping install - assigning regions to regionservers - recovering from regionserver failures

HBase made up of an HBase master node orchestrating a cluster of one or more regionserver workers

• RegionServer'sroles: - manages region splits - informing master about changes

• HBase depends on ZooKeeper, and by default it manages a ZooKeeper instance as the authority on cluster state

When to use HBase

• You need random write, random read or both (but not neither)

• You need to do many thousands of operations per sec on multiple TB of data

• Your access patterns are simple

adv of Hbase

-no sepaarate storage

-super fast lokup by using various levels of indexing

-store version of data

-good for high transaction. eg. millions of events / hour

disv

-two level of master (name node, HMaster)

single point of failure

**cassandra**

The coordinator also applies the Consistency Level (CL)

• Consistency level (CL) – how many nodes must acknowledge a read or write request

• CL may vary for each request

• On success, coordinator notifies client

• Possible consistency levels include

• ANY • ONE • QUORUM ( RF / 2 ) + 1 • ALL

What is the partitioner? • A system on each node which hashes tokens from designated values in rows being added

Row is the smallest unit that stores related data in Cassandra

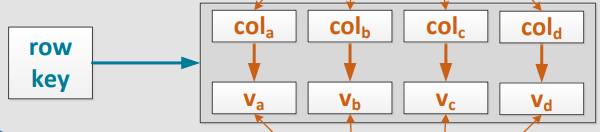
• Rows – individual rows constitute a column family

• Row key – uniquely identifies a row in a column family

• Row – stores pairs of column keys and column values

• Column key – uniquely identifies a column value in a row

• Column value – stores one value or a collection of values



partitioning key

key on basis of which we physically partition data

data with same value of partition key is stored together in one node

can be composite

eg.Tx + DFW, TX+HOU

within each partition, for faster look up -> clustering column

primary Key (partition Key + another key) == (part1, part2)

can only query with all partition-key colomun