high volume, velocity and variety, volatility (constantly changing patterns so ever unstable)

Apache Hadoop software:
open source framework for distributed computing
for the distributed storage &
processing of large datasets stored on HDFS across clusters of computers
using simple programming models for data processing like mapreduce

kafka open source streaming paltform

mapreduce takes in large volumes of data, divides it into chunks, parallely processing em on servers and then aggregates them in results

binge watch karhe? mostly konse duration ke karhe? pause karke resume kiya ya ditch? watching patterns kya h? breakup ke baad love movies ya emo hi? thumbnail personalization

==========================mod5==
dsms - data stream management systems, has a bounded moemory
dsm - data stream mining
system cannot store entire data stream since u dont have an infinite storage
1. its in sm: adhoc, mm: standing queries
2. u store approximate summary structures.
3. u sample ur stream data

thus query shouldnt require the entire stream as input at once. order based (as stored in sm) and time based (as they arrive) queries should be allowed. backtracking/ needing the old data again, isnt feasible (ONLY ONE PASS OVER DATA). parallel and shared execution of these continuous queries must be possible.

data stream model
query running continuously as the data ip changes, thus returns new results as the new data arrives

applications:
how many miss-clicks: gihtub vs github
number of clicks on a website today? yest
trends on twitter

concept drift:
the underlying concepts or patterns in the data are not static; they evolve or drift. thus change reqd in use of algos that were trained on historic data

continuous flow of data, dynamic, infinite, changes, fast changes and speed
Needs multidim processing
n/w traffic monitoring , congestion
social media analytics
trading finance (forecast future valuation holding or selling the shares that u own)

One-Time Queries:
- One-time queries involve performing a single analysis or task on a historical dataset or data stream, typically for a specific, non-recurring purpose. Once the query is executed, it doesn't continue to monitor the data stream or provide ongoing results. It's a one-off operation. For example, calculating the average temperature for a specific month is a one-time query. It's done once, and you get the result, and it's not continuously updated.

Continuous Queries:
- Continuous queries, on the other hand, are designed to continuously monitor and analyze data streams in real-time. They are ongoing, repetitive tasks that provide real-time or near-real-time results. These queries are set up to process incoming data as it arrives and continually update their outputs. For example, continuously monitoring the real-time location of delivery trucks using GPS data involves a continuous query that updates the truck's locations as new data becomes available.

In summary, one-time queries are performed once and don't continuously update their results, while continuous queries operate on real-time data streams and provide ongoing, updated results.

Type of query in data stream mining:

One-Time Queries:
- Calculating the average daily temperature over the past month from a historical weather data stream.

Continuous Queries:
- Tracking the real-time location of delivery trucks using GPS data from a continuous stream.

Adhoc Queries:
- Checking the current price of a specific product on an e-commerce website at any time.

Predefined Queries:
- Setting an alert to be notified when a stock's price falls below a certain threshold in a real-time stock market data stream.

Cloaking

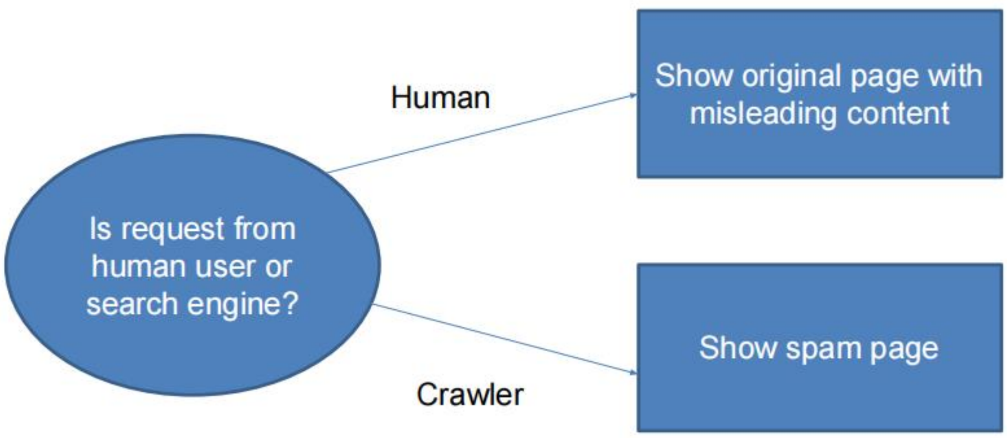
Returning different pages to Search Engine than what is returned to the people.

Doorway Pages

Low quality WebPages (dummy pages) created with very little content, but stuffed with very similar keywords and phrases.

They redirect the user to a page containing content of a more commercial nature

Cloaking


Human
Show original page with misleading content
Is request from human user or search engine?
Show spam page
Crawler

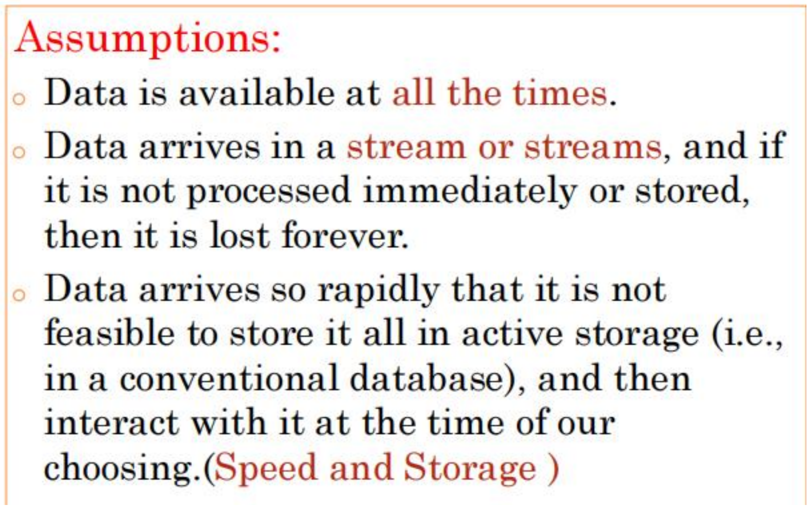
Today's applications needs Agile system that can process unstructured real time data dynamically with flexible schema, higly available, rich queried, open source, fault tolerant due to replication, partitionable, scale-out karskte
not provided:
 sql, acid, group by, joins

Centralized and distributed
Nosql db
Sql relational db

key value: for relationed data, session info (uid1:{sanjana,01,d17b} AMAZON DYnaMO DB
colmn: log record transactions   CASSANDRA HBASE
doc: blogging MONGODB
graph: social nw.
Redis: hierarchical: n/w nodes
Oop db

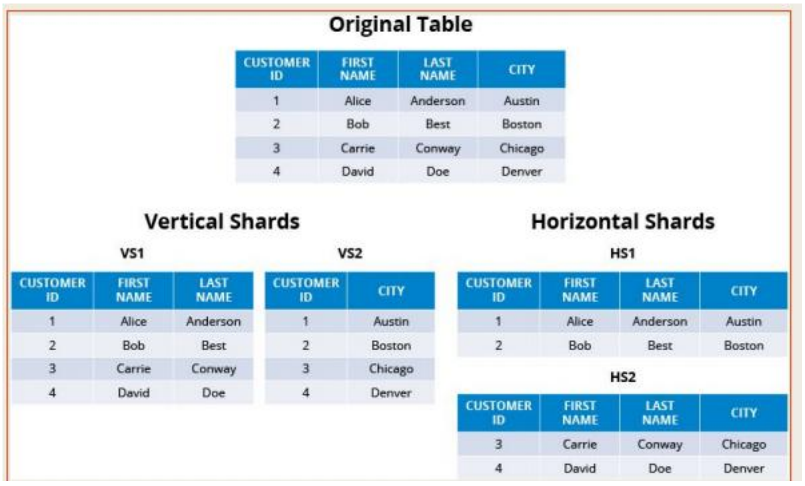
personalization in rs: 3 Rs
right time, right offer, right customer

Consistent Hashing: Consistent hashing is a technique used to distribute data across nodes in a way that minimizes the need for data migration when nodes are added or removed. It ensures that data is uniformly distributed and minimizes hotspots.



Assumptions:

- Data is available at all the times.
- Data arrives in a stream or streams, and if it is not processed immediately or stored, then it is lost forever.
- Data arrives so rapidly that it is not feasible to store it all in active storage (i.e., in a conventional database), and then interact with it at the time of our choosing.(Speed and Storage)



Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Vertical Shards

VS1

CUSTOMER ID	FIRST NAME	LAST NAME
1	Alice	Anderson
2	Bob	Best
3	Carrie	Conway
4	David	Doe

VS2

CUSTOMER ID	CITY
1	Austin
2	Boston
3	Chicago
4	Denver

Horizontal Shards

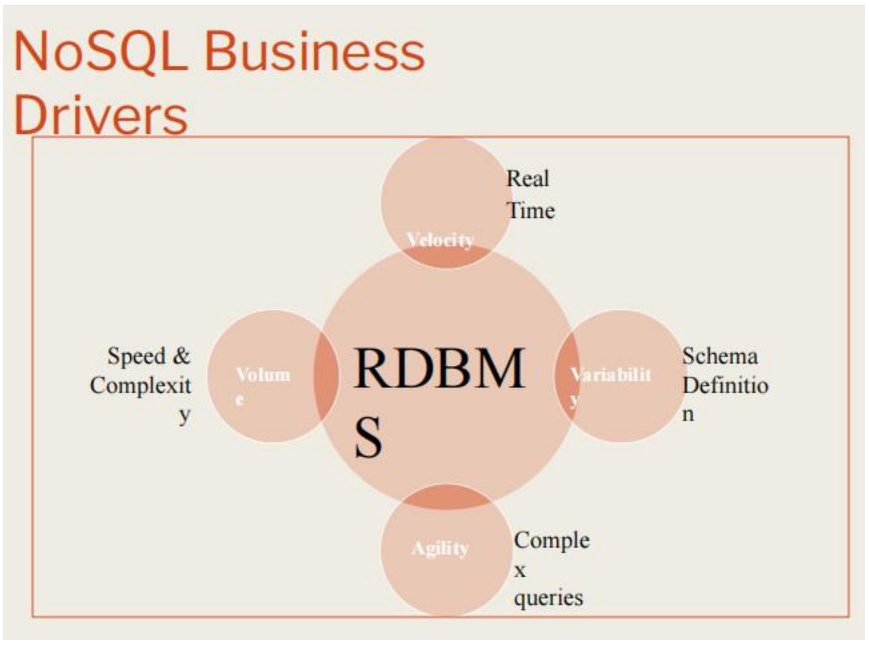
HS1

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston

HS2

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Vertical shard is a new database schema
Horizontal shards is a sub dataset
Shards(partitions) are wrt some index.



cloud database/ dbaas where data is stored in a virtual environment and executes over the cloud computing platform, accesible thru internet, provides saas, paas, etc. aws azure

Databases are no longer one-size-fits-all database no more!!

SPoF can be avoided by using RAID drives or by using a standby master that is continuously updated by the master node. Hbase uses master slave model

An edge always has a start node, end node, type, and direction, and an edge can describe parent-child relationships, actions, ownership, and the like. Every node and edge has a unique identifier.

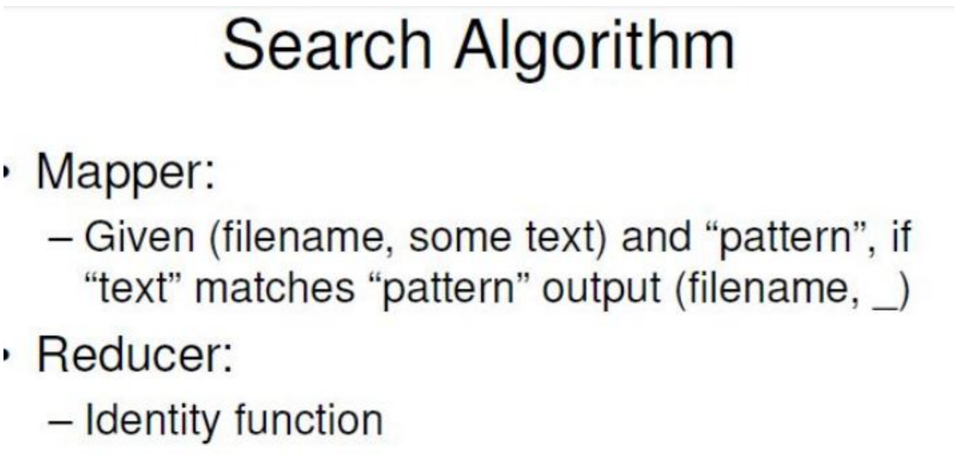
Records in a MongoDB database are called documents, and the field values
may include numbers, strings, booleans, arrays, or even nested
documents. A MongoDB deployment hosts a number of
databases.
■ A database holds a set of collections.
■ A collection holds a set of documents.
■ A document is a set of key-value pairs.
data associated with a record is stored within the same document.
dob certi not given: empty field vs not in doc

==

ef cod for sql
carl Strozz for nosql

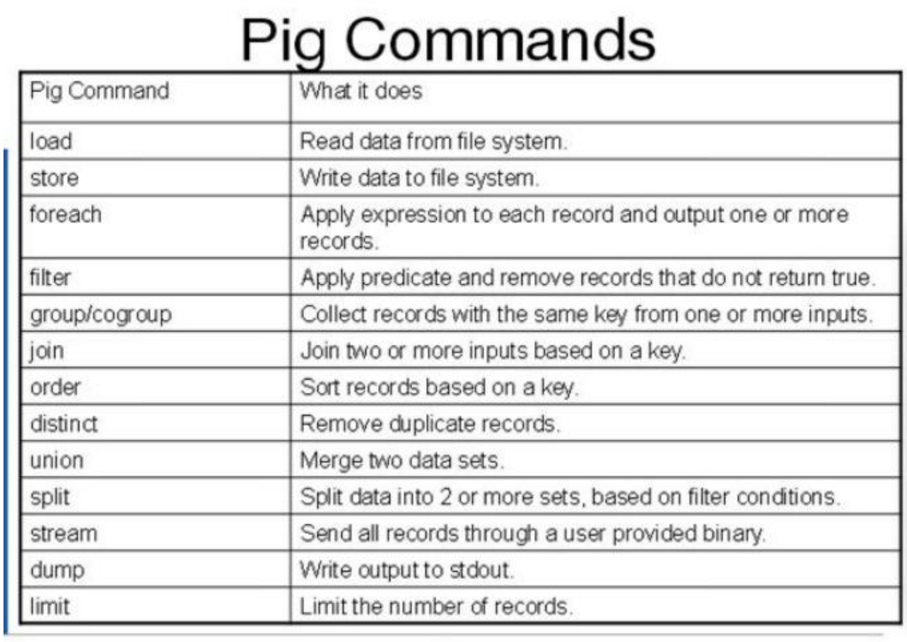
nosql data architecture patterns: 4 types
brewers cap thm
business drivers for nosql: agility, variability, velocity, volume

scale out (also known as horizontal scaling), rather than scale up (faster processors), Scale up" our systems by upgrading our existing hardware. This process is expensive.



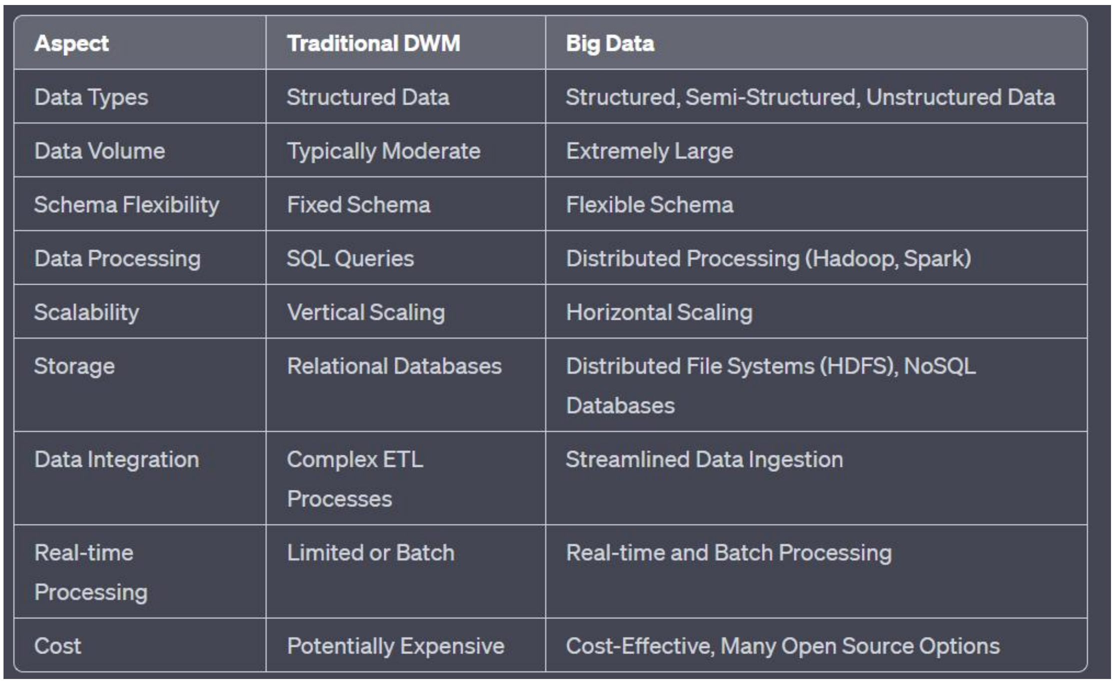
Search Algorithm

- Mapper:
  - Given (filename, some text) and “pattern”, if “text” matches “pattern” output (filename, _)
- Reducer:
  - Identity function



Pig Commands

Pig Command	What it does
load	Read data from file system.
store	Write data to file system.
foreach	Apply expression to each record and output one or more records.
filter	Apply predicate and remove records that do not return true.
group/cogroup	Collect records with the same key from one or more inputs.
join	Join two or more inputs based on a key.
order	Sort records based on a key.
distinct	Remove duplicate records.
union	Merge two data sets.
split	Split data into 2 or more sets, based on filter conditions.
stream	Send all records through a user provided binary.
dump	Write output to stdout.
limit	Limit the number of records.



Aspect	Traditional DWM	Big Data
Data Types	Structured Data	Structured, Semi-Structured, Unstructured Data
Data Volume	Typically Moderate	Extremely Large
Schema Flexibility	Fixed Schema	Flexible Schema
Data Processing	SQL Queries	Distributed Processing (Hadoop, Spark)
Scalability	Vertical Scaling	Horizontal Scaling
Storage	Relational Databases	Distributed File Systems (HDFS), NoSQL Databases
Data Integration	Complex ETL Processes	Streamlined Data Ingestion
Real-time Processing	Limited or Batch	Real-time and Batch Processing
Cost	Potentially Expensive	Cost-Effective, Many Open Source Options

Artificial Linking: Link farms consist of websites or pages that have little or no relevance to each other. They are created with the sole purpose of exchanging links, often in a reciprocal manner, without considering the quality or relevance of the content.

HADOOP ECOSYSTEM:

1. Core Hadoop ecosystem is nothing but the different components that are built on the Hadoop platform directly.
2. Figure 1.9 represents Hadoop Ecosystem.

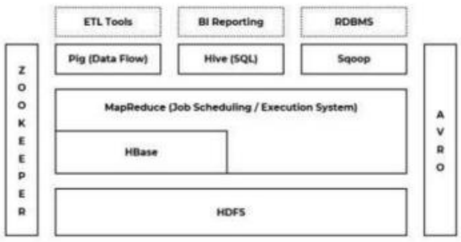


Figure 1.9: Hadoop Ecosystem.

I) Hadoop Distributed File System (HDFS):

1. **HDFS is the foundation of Hadoop** and hence is a very important component of the Hadoop ecosystem.
2. It is Java software that provides many features like scalability, high availability, fault tolerance, cost effectiveness etc.
3. It also provides robust distributed data storage for Hadoop.
4. We can deploy many other software frameworks over HDFS.

II) MapReduce:

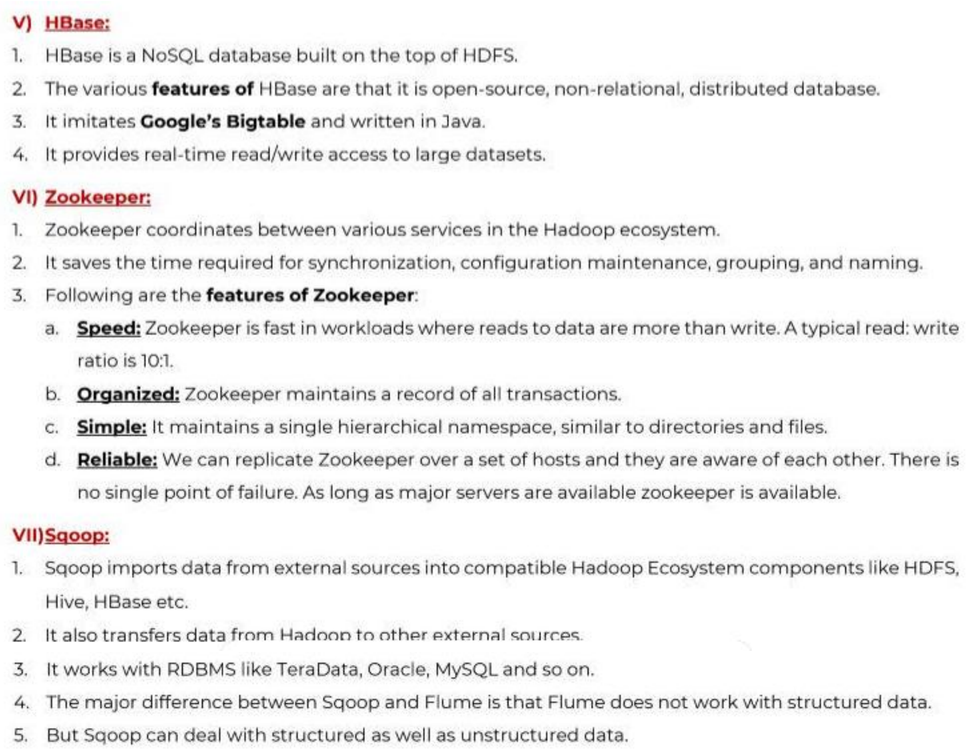
1. **MapReduce** is the data processing component of Hadoop.
2. It applies the computation on sets of data in parallel thereby improving the performance.
3. MapReduce works in two phases:
   a. **Map Phase:** This phase takes input as key-value pairs and produces output as key-value pairs. It can write custom business logic in this phase. Map phase processes the data and gives it to the next phase.
   b. **Reduce Phase:** The MapReduce framework sorts the key-value pair before giving the data to this phase. This phase applies the summary type of calculations to the key-value pairs.

III) Hive:

1. Hive is a data warehouse project built on the top of Apache Hadoop which provides data query and analysis.
2. It has got the language of its own call HQL or **Hive Query Language**.
3. HQL automatically translates the queries into the corresponding map-reduce job.
4. Main parts of the Hive are –
   a. **MetaStore:** It stores metadata
   b. **Driver:** Manages the lifecycle of HQL statement
   c. **Query Compiler:** Compiles HQL into DAG i.e. Directed Acyclic Graph
   d. **Hive Server:** Provides interface for JDBC/ODBC server.

IV) Pig:

1. Pig is a SQL like language used for querying and analyzing data stored in HDFS.
2. Yahoo was the original creator of the Pig.
3. It uses pig latin language.
4. It loads the data, applies a filter to it and dumps the data in the required format.
5. Pig also consists of JVM called Pig Runtime. Various **features of Pig** are as follows:-
   a. **Extensibility:** For carrying out special purpose processing, users can create their own custom function.
   b. **Optimization opportunities:** Pig automatically optimizes the query allowing users to focus on semantics rather than efficiency.
   c. **Handles all kinds of data:** Pig analyzes both structured as well as unstructured.



V) HBase:

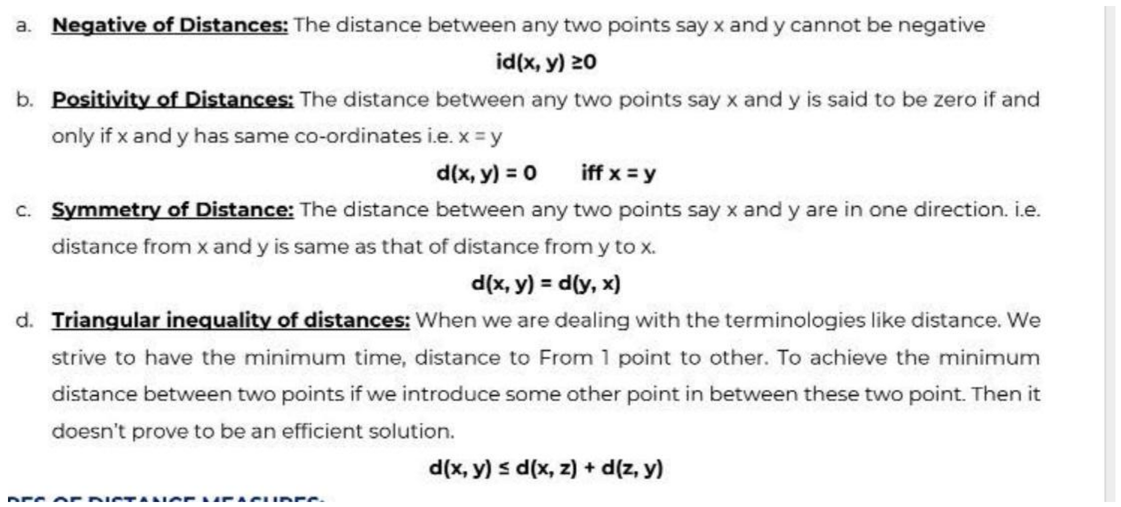
1. HBase is a NoSQL database built on the top of HDFS.
2. The various **features of** HBase are that it is open-source, non-relational, distributed database.
3. It imitates **Google's Bigtable** and written in Java.
4. It provides real-time read/write access to large datasets.

VI) Zookeeper:

1. Zookeeper coordinates between various services in the Hadoop ecosystem.
2. It saves the time required for synchronization, configuration maintenance, grouping, and naming.
3. Following are the **features of Zookeeper**:
   a. **Speed:** Zookeeper is fast in workloads where reads to data are more than write. A typical read: write ratio is 10:1.
   b. **Organized:** Zookeeper maintains a record of all transactions.
   c. **Simple:** It maintains a single hierarchical namespace, similar to directories and files.
   d. **Reliable:** We can replicate Zookeeper over a set of hosts and they are aware of each other. There is no single point of failure. As long as major servers are available zookeeper is available.

VII)Sqoop:

1. Sqoop imports data from external sources into compatible Hadoop Ecosystem components like HDFS, Hive, HBase etc.
2. It also transfers data from Hadoop to other external sources.
3. It works with RDBMS like TeraData, Oracle, MySQL and so on.
4. The major difference between Sqoop and Flume is that Flume does not work with structured data.
5. But Sqoop can deal with structured as well as unstructured data.



a. **Negative of Distances:** The distance between any two points say x and y cannot be negative

$$id(x, y) \geq 0$$

b. **Positivity of Distances:** The distance between any two points say x and y is said to be zero if and only if x and y has same co-ordinates i.e. x = y

$$d(x, y) = 0 \qquad \text{iff } x = y$$

c. **Symmetry of Distance:** The distance between any two points say x and y are in one direction. i.e. distance from x and y is same as that of distance from y to x.

$$d(x, y) = d(y, x)$$

d. **Triangular inequality of distances:** When we are dealing with the terminologies like distance. We strive to have the minimum time, distance to From 1 point to other. To achieve the minimum distance between two points if we introduce some other point in between these two point. Then it doesn't prove to be an efficient solution.

$$d(x, y) \leq d(x, z) + d(z, y)$$

PES OF DISTANCE MEASURES:

In Hadoop, handling failures is a crucial aspect of ensuring the reliability and fault tolerance of a distributed computing system. Hadoop employs several mechanisms to manage failures, including hardware failures, software failures, and other issues that can occur in a cluster. Here are some key methods for handling failures in Hadoop:

1. **Data Replication**:
   - Hadoop stores data in a distributed file system called Hadoop Distributed File System (HDFS). To handle data node failures, HDFS replicates data across multiple data nodes. By default, each block of data is replicated three times. If a data node fails, the system can retrieve the data from one of the replicas on another data node.

2. **Heartbeats and Health Checks**:
   - Hadoop employs a heartbeat mechanism to detect the health of data nodes and task trackers (in the MapReduce component). Data nodes and task trackers send periodic heartbeats to the master nodes (NameNode and JobTracker), indicating that

they are alive and functioning. If a node stops sending heartbeats, it is considered failed, and data is automatically redistributed to healthy nodes.

3. **Task Redundancy**:
   - In the MapReduce framework, task trackers can fail during the execution of a Map or Reduce task. JobTracker monitors task progress and reschedules tasks on other nodes if a task tracker fails. This redundancy ensures that tasks are eventually completed even if some task trackers fail.

4. **Secondary NameNode**:
   - HDFS includes a secondary NameNode that helps in periodically merging the edit logs with the main namespace image. This process reduces the recovery time required in the event of a NameNode failure.

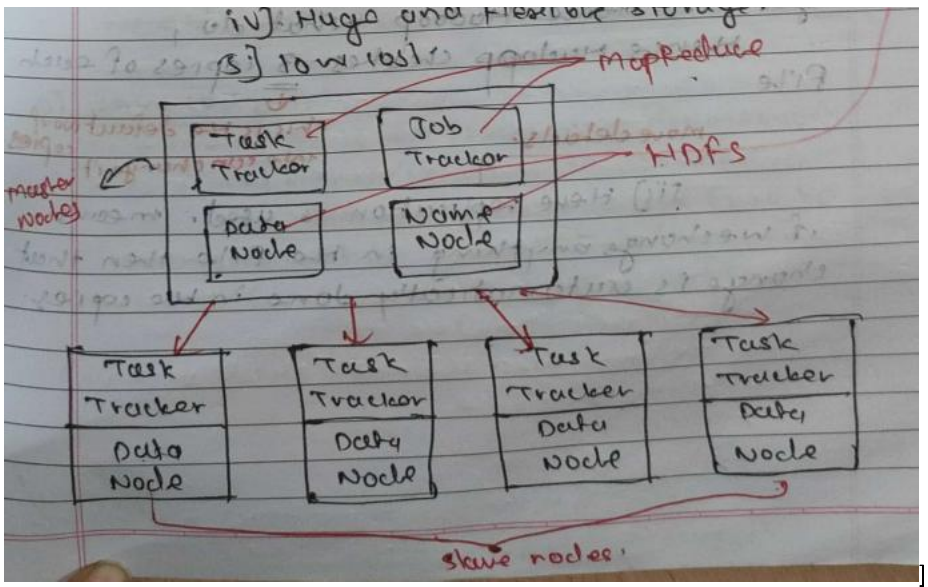
5. **Checkpointing**:
   - Hadoop performs periodic checkpointing to create a consistent snapshot of the file system metadata. This snapshot can be used to recover from a failed NameNode by restoring the file system's metadata to a previous state.

7. **Monitoring and Logging**:
   - Hadoop provides extensive logging and monitoring capabilities to help administrators and operators identify issues and troubleshoot failures. Logs contain valuable information for diagnosing problems.

8. **Rack Awareness**:
   - Hadoop is rack-aware, meaning it understands the physical topology of the data nodes in a cluster. This knowledge helps optimize data placement and reduce the impact of network failures.

Handling failures is a fundamental aspect of Hadoop's design, and these mechanisms are in place to ensure that the system can recover from various types of failures and continue to operate reliably in a distributed computing environment.

Concept Drift: Data streams are dynamic, and the underlying data distribution may change over time. Clustering models must be adaptive and able to detect and respond to concept drift by updating clusters as the data evolves.



]

triangular matrix in main memory counting? pcy algo. Park chen yu

hdfs and mr are major components of hadoop
mahout mc learning .. recommendation classfin clustering

zookepeer: allow inter communication and stop interrupts in processing

ordered pair of nodes
weighted direction edges
relationships
in degree out degree
sng mining: largest smallest dist bw 2 nodes, clusters, density/concentrated
geodesic dist: actual dist bw 2 nodes

dce- distributed computing environment. thrashing : too much time wasted in fetching data from sm rather than execution

social nws: collaborative (badminton), telephone, email
simrank: rank bw similar nodes for random walkers

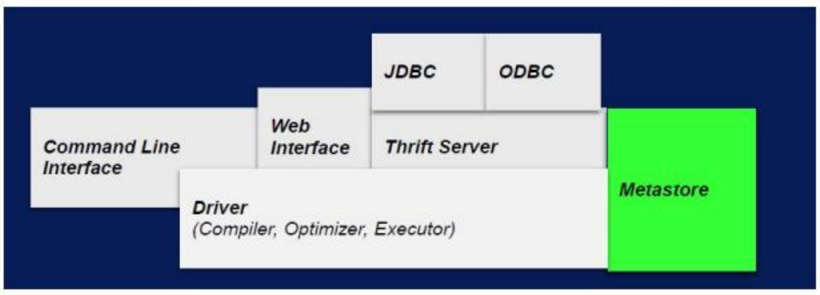
u search smth, web crawlers get u a list of matching docs: inverted index data structure
every term that appears in here, w some page rank associated, from this inverted index is extracted and searched for

#back links: indegree
#fwd links: outdegree

property of having the sum of page ranks being 1 is called STOCHASTICITY
so either row full being 0 before transformation of adjacency matrix. or colm full 0 if M

TELEPORT/DAMPING FACTOR

HIVE Architecture



Hive is a data warehousing and SQL-like query language tool developed by Facebook, which is now part of the Apache Software Foundation. It is designed for querying and managing large datasets in a distributed storage system, typically Hadoop Distributed File System (HDFS). Hive provides a high-level interface for querying data stored in Hadoop, making it accessible to users who are familiar with SQL.

Here's an overview of Hive and its purpose:

Key Features and Components of Hive:

1. **Hive Query Language (HQL)**: Hive uses a SQL-like query language known as HQL to express queries and data transformations. Users can write HQL queries to process, filter, aggregate, and analyze data stored in HDFS.

2. **Data Warehousing**: Hive is often used for data warehousing, providing tools to create, store, and manage large-scale datasets efficiently.

3. **Schema on Read**: Unlike traditional relational databases, which use a schema-on-write approach, Hive follows a "schema on read" model. This means that data is stored in its raw form in HDFS, and schema information is applied when the data is read. This allows for flexibility in handling semi-structured or unstructured data.

4. **Hive Metastore**: Hive includes a metastore, which serves as a centralized repository for metadata about tables, columns, data types, and storage location. This makes it easy to manage metadata and share it across different Hive instances.

5. **Integration with Hadoop**: Hive is tightly integrated with the Hadoop ecosystem, and it can leverage the distributed processing capabilities of Hadoop, including MapReduce, Tez, and Spark, to execute queries.

Why Hive is Needed:
1. **SQL-Like Query Language**: Hive provides a familiar SQL-like query language, making it accessible to users who are already skilled in SQL. This lowers the barrier to entry for working with big data technologies like Hadoop.

2. **Scalability**: Hive is designed for processing and querying large datasets stored in HDFS. It can handle petabytes of data, making it suitable for big data processing.

3. **Schema Flexibility**: Hive's schema-on-read approach allows users to work with data in a flexible manner. It's well-suited for situations where the structure of the data is not known in advance or can change over time.

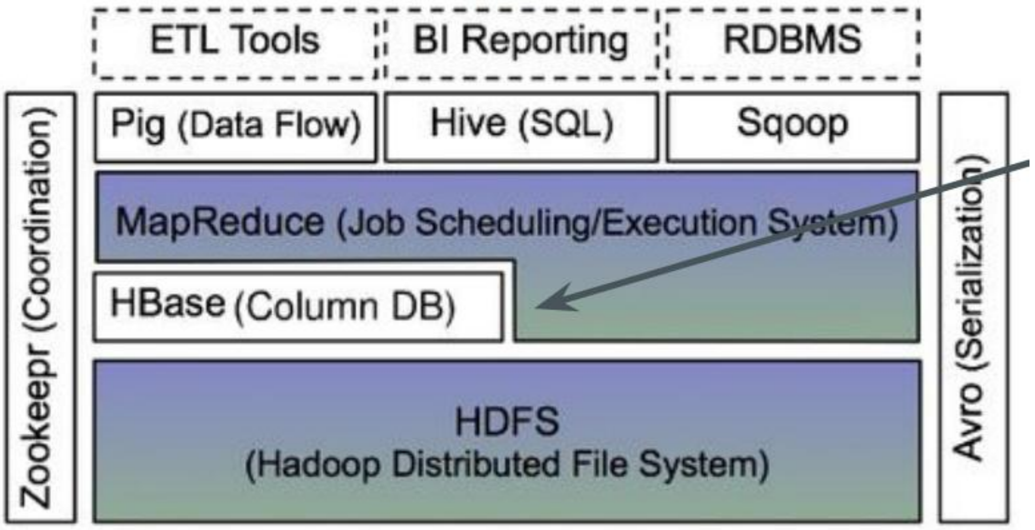
4. **Hadoop Ecosystem Integration**: Hive seamlessly integrates with other Hadoop ecosystem tools and frameworks, enabling organizations to build end-to-end data processing pipelines.

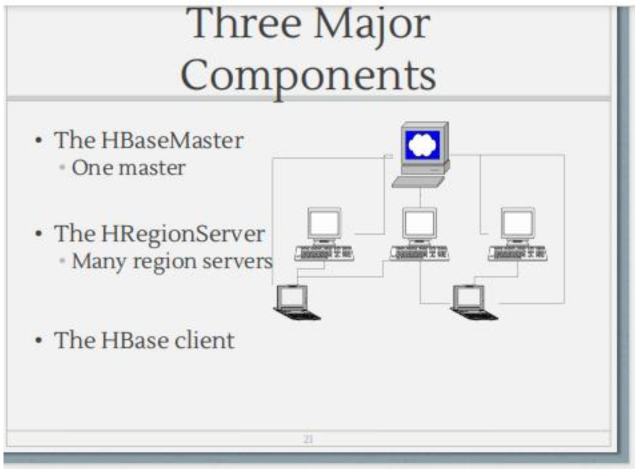
5. **Centralized Metadata Management**: The Hive Metastore simplifies metadata management, making it easier to catalog and organize large datasets.

6. **Extensibility**: Hive can be extended through user-defined functions (UDFs) and custom serializers/deserializers to support various data formats and processing requirements.

Overall, Hive is a valuable tool for organizations working with large-scale, distributed datasets in Hadoop. It provides a SQL-like interface for data processing and allows users to leverage the power of Hadoop's distributed computing capabilities.

The Hadoop Ecosystem

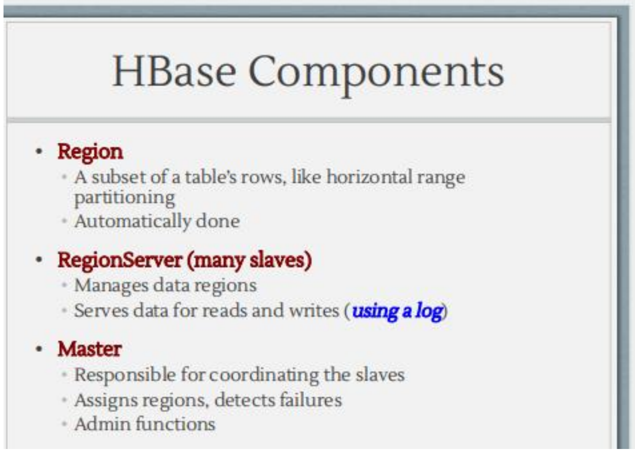




Three Major Components

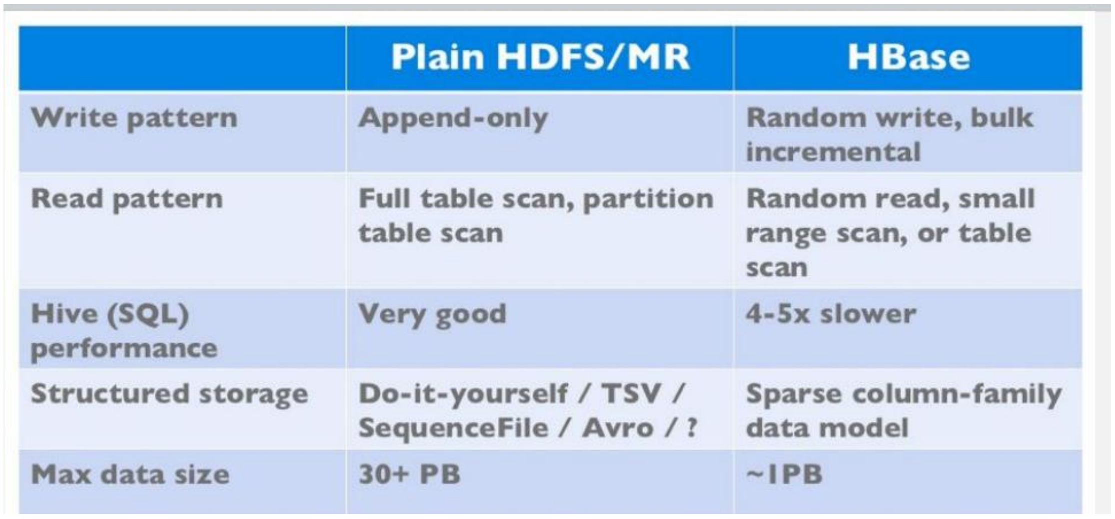
- The HBaseMaster
  - One master
- The HRegionServer
  - Many region servers
- The HBase client

21



HBase Components

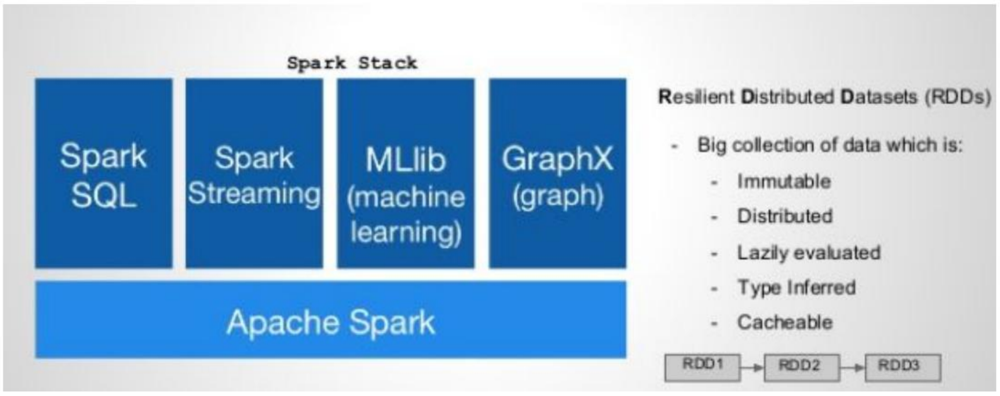
- **Region**
  - A subset of a table's rows, like horizontal range partitioning
  - Automatically done
- **RegionServer (many slaves)**
  - Manages data regions
  - Serves data for reads and writes (***using a log***)
- **Master**
  - Responsible for coordinating the slaves
  - Assigns regions, detects failures
  - Admin functions



	Plain HDFS/MR	HBase
Write pattern	Append-only	Random write, bulk incremental
Read pattern	Full table scan, partition table scan	Random read, small range scan, or table scan
Hive (SQL) performance	Very good	4-5x slower
Structured storage	Do-it-yourself / TSV / SequenceFile / Avro / ?	Sparse column-family data model
Max data size	30+ PB	~1PB

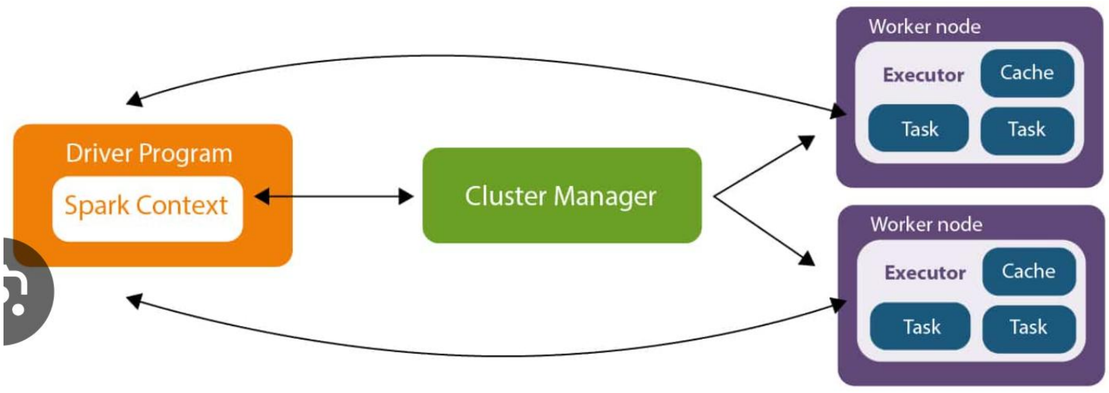
HBase is better for specific use cases where scalability, high write throughput, and schema flexibility are essential. It excels in scenarios involving big data, time-series data, or situations where traditional RDBMS may struggle to manage the sheer volume and variety of data. Its distributed architecture and suitability for real-time processing make it a strong choice for big data applications. However, it may not be the best choice for transactional or complex relational data with many interrelated tables.

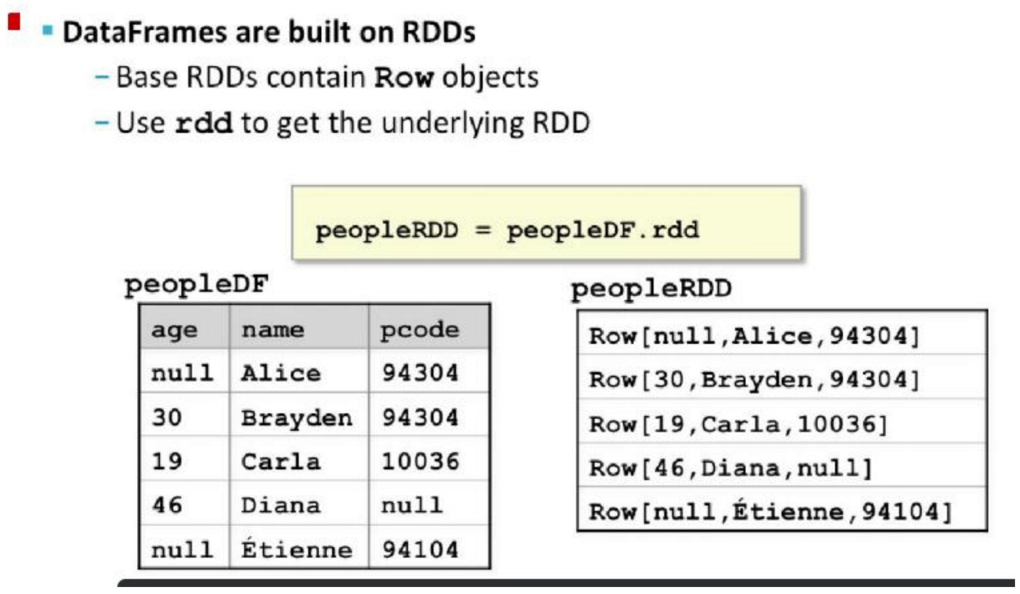
PYSPARK



PySpark is an open-source, distributed data processing framework that integrates the Python programming language with Apache Spark. It provides an efficient and versatile platform for processing large-scale data, offering tools for data manipulation, analytics, machine learning, and distributed computing across clusters, making big data analysis more accessible to Python developers.

FOR ML AND sQL





- **DataFrames are built on RDDs**
  - Base RDDs contain **Row** objects
  - Use **rdd** to get the underlying RDD

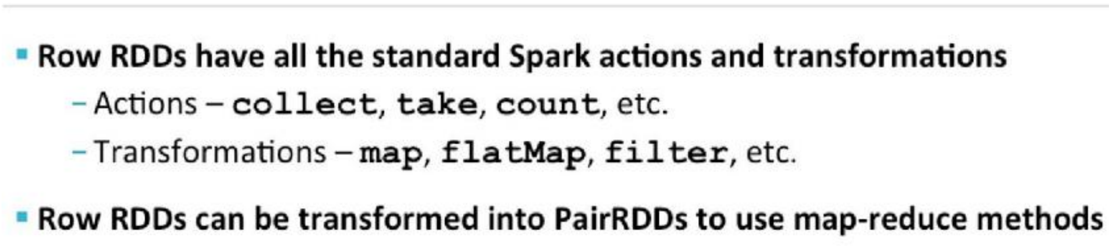
```
peopleRDD = peopleDF.rdd
```

peopleDF

age	name	pcode
null	Alice	94304
30	Brayden	94304
19	Carla	10036
46	Diana	null
null	Étienne	94104

peopleRDD

```
Row[null,Alice,94304]
Row[30,Brayden,94304]
Row[19,Carla,10036]
Row[46,Diana,null]
Row[null,Étienne,94104]
```



- **Row RDDs have all the standard Spark actions and transformations**
  - Actions – **collect**, **take**, **count**, etc.
  - Transformations – **map**, **flatMap**, **filter**, etc.
- **Row RDDs can be transformed into PairRDDs to use map-reduce methods**

Some Common transformations

- **map**(function)- **creates a new RDD** by performing a function on each record in the base RDD.
- **filter**(function) - **creates a new RDD by including or excluding** each record in the base RDD to a boolean function.