INDEX

V	(Big Data Frame Works for Analytics) Hadoop Frame Work: Map Reduce Programming: I/O formats, Map side join- Reduce Side Join-Secondary Sorting- Pipelining Map Reduce jobs	72-78
	Spark Frame Work: Introduction to Apache spark	
	How spark works, Programming with RDDs: Create RDD spark Operations-Data Frame.	

UNIT - I

Syllabus:

Introduction to big data: Data, Characteristics of data and Types of digital data: Unstructured, Semi-structured and Structured - Sources of data. Big Data Evolution-Definition of big data-Characteristics and Need of big data-Challenges of .Big data analytics, Overview of business intelligence.

What is Data?

Data is defined as individual facts, such as numbers, words, measurements, observations or just descriptions of things.

For example, data might include individual prices, weights, addresses, ages, names, temperatures, dates, or distances.

There are two main types of data:

1. **Quantitative data** is provided in numerical form, like the weight, volume, or cost of an item.
2. **Qualitative data** is descriptive, but non-numerical, like the name, sex, or eye colour of a person.

Characteristics of Data

The following are six key characteristics of data which discussed below:

1. Accuracy
2. Validity
3. Reliability
4. Timeliness
5. Relevance
6. Completeness

1. Accuracy

Data should be sufficiently accurate for the intended use and should be captured only once, although it may have multiple uses. Data should be captured at the point of activity.

2. Validity

Data should be recorded and used in compliance with relevant requirements, including the correct application of any rules or definitions. This will ensure consistency between periods and with similar organizations, measuring what is intended to be measured.

3. Reliability

Data should reflect stable and consistent data collection processes across collection points and over time. Progress toward performance targets should reflect real changes rather than variations in data collection approaches or methods. Source data is clearly identified and readily available from manual, automated, or other systems and records.

4. Timeliness

Data should be captured as quickly as possible after the event or activity and must be available for the intended use within a reasonable time period. Data must be available quickly and frequently enough to support information needs and to influence service or management decisions.

5. Relevance

Data captured should be relevant to the purposes for which it is to be used. This will require a periodic review of requirements to reflect changing needs.

6. Completeness

Data requirements should be clearly specified based on the information needs of the organization and data collection processes matched to these requirements.

Types of Digital Data

- Digital data is the electronic representation of information in a format or language that machines can read and understand.
- In more technical terms, Digital data is a binary format of information that's converted into a machine-readable digital format.
- The power of digital data is that any analog inputs, from very simple text documents to

genome sequencing results, can be represented with the binary system.

Types of Digital Data:

- Structured
- Unstructured
- Semi Structured Data

Structured Data:

- Structured data refers to any data that resides in a fixed field within a record or file.
- Having a particular Data Model.
- Meaningful data.
- Data arranged in arow and column.
- Structured data has the advantage of being easily entered, stored, queried and analysed.
- E.g.: Relational Data Base, Spread sheets.
- Structured data is often managed using Structured Query Language (SQL)

Sources of Structured Data:

- SQL Databases
- Spreadsheets such as Excel
- OLTP Systems
- Online forms
- Sensors such as GPS or RFID tags
- Network and Web server logs
- Medical devices

Advantages of Structured Data:

- Easy to understand and use: Structured data has a well-defined schema or data model, making it easy to understand and use. This allows for easy data retrieval, analysis, and reporting.
- Consistency: The well-defined structure of structured data ensures consistency and accuracy in the data, making it easier to compare and analyze data across different sources.

- Efficient storage and retrieval: Structured data is typically stored in relational databases, which are designed to efficiently store and retrieve large amounts of data. This makes it easy to access and process data quickly.
- Enhanced data security: Structured data can be more easily secured than unstructured or semi-structured data, as access to the data can be controlled through database security protocols.
- Clear data lineage: Structured data typically has a clear lineage or history, making it easy to track changes and ensure data quality.

Disadvantages of Structured Data:

- Inflexibility: Structured data can be inflexible in terms of accommodating new types of data, as any changes to the schema or data model require significant changes to the database.
- Limited complexity: Structured data is often limited in terms of the complexity of relationships between data entities. This can make it difficult to model complex real-world scenarios.
- Limited context: Structured data often lacks the additional context and information that unstructured or semi-structured data can provide, making it more difficult to understand the meaning and significance of the data.
- Expensive: Structured data requires the use of relational databases and related technologies, which can be expensive to implement and maintain.
- Data quality: The structured nature of the data can sometimes lead to missing or incomplete data, or data that does not fit cleanly into the defined schema, leading to data quality issues.

Unstructured Data:

- Unstructured data can not readily classify and fit into a neat box
- Also called unclassified data.
- Which does not confirm to any data model.
- Business rules are not applied.
- Indexing is not required.

- E.g.: photos and graphic images, videos, streaming instrument data, webpages, Pdf files, PowerPoint presentations, emails, blog entries, wikis and word processing documents.

Sources of Unstructured Data:

- Web pages
- Images (JPEG, GIF, PNG, etc.)
- Videos
- Memos
- Reports
- Word documents and PowerPoint presentations
- Surveys

Advantages of Unstructured Data:

- Its supports the data which lacks a proper format or sequence
- The data is not constrained by a fixed schema
- Very Flexible due to absence of schema.
- Data is portable
- It is very scalable
- It can deal easily with the heterogeneity of sources.
- These type of data have a variety of business intelligence and analytics applications.

Disadvantages Of Unstructured data:

- It is difficult to store and manage unstructured data due to lack of schema and structure
- Indexing the data is difficult and error prone due to unclear structure and not having pre-defined attributes. Due to which search results are not very accurate.
- Ensuring security to data is difficult task.

Semi structured Data:

- Self-describing data.

- Metadata (Data about data).
- Also called quiz data: data in between structured and semi structured.
- It is a type of structured data but not followed data model.
- Data which does not have rigid structure.
- E.g.: E-mails, word processing software.
- XML and other markup language are often used to manage semi structured data.

Sources of semi-structured Data:

- E-mails
- XML and other markup languages
- Binary executables
- TCP/IP packets
- Zipped files
- Integration of data from different sources
- Web pages

Advantages of Semi-structured Data:

- The data is not constrained by a fixed schema
- Flexible i.e Schema can be easily changed.
- Data is portable
- It is possible to view structured data as semi-structured data
- Its supports users who can not express their need in SQL
- It can deal easily with the heterogeneity of sources.
- Flexibility: Semi-structured data provides more flexibility in terms of data storage and management, as it can accommodate data that does not fit into a strict, predefined schema. This makes it easier to incorporate new types of data into an existing database or data processing pipeline.
- Scalability: Semi-structured data is particularly well-suited for managing large volumes of data, as it can be stored and processed using distributed computing systems, such as Hadoop or Spark, which can scale to handle massive amounts of data.

- Faster data processing: Semi-structured data can be processed more quickly than traditional structured data, as it can be indexed and queried in a more flexible way. This makes it easier to retrieve specific subsets of data for analysis and reporting.
- Improved data integration: Semi-structured data can be more easily integrated with other types of data, such as unstructured data, making it easier to combine and analyze data from multiple sources.
- Richer data analysis: Semi-structured data often contains more contextual information than traditional structured data, such as metadata or tags. This can provide additional insights and context that can improve the accuracy and relevance of data analysis.

Disadvantages of Semi-structured data

- Lack of fixed, rigid schema make it difficult in storage of the data
- Interpreting the relationship between data is difficult as there is no separation of the schema and the data.
- Queries are less efficient as compared to structured data.
- Complexity: Semi-structured data can be more complex to manage and process than structured data, as it may contain a wide variety of formats, tags ,and metadata. This can make it more difficult to develop and maintain data models and processing pipelines.
- Lack of standardization: Semi-structured data often lacks the standardization and consistency of structured data, which can make it more difficult to ensure data quality and accuracy. This can also make it harder to compare and analyze data across different sources.
- Reduced performance: Processing semi-structured data can be more resource-intensive than processing structured data, as it often requires more complex parsing and indexing operations. This can lead to reduced performance and longer processing times.
- Limited tooling: While there are many tools and technologies available for working with structured data, there are fewer options for working with semi- structured data. This can make it more challenging to find the right tools and technologies for a particular use case.

- Data security: Semi-structured data can be more difficult to secure than structured data, as it may contain sensitive information in unstructured or less-visible parts of the data. This can make it more challenging to identify and protect sensitive information from unauthorized access.
- Overall, while semi-structured data offers many advantages in terms of flexibility and scalability, it also presents some challenges and limitations that need to be carefully considered when designing and implementing data processing and analysis pipelines.

Big Data

Big Data **is** a collection of data that is huge in volume, yet growing exponentially with time. It is a data with so large size and complexity that none of traditional data management tools can store it or process it efficiently. Big data is also a data but with huge size.

What is an Example of Big Data?

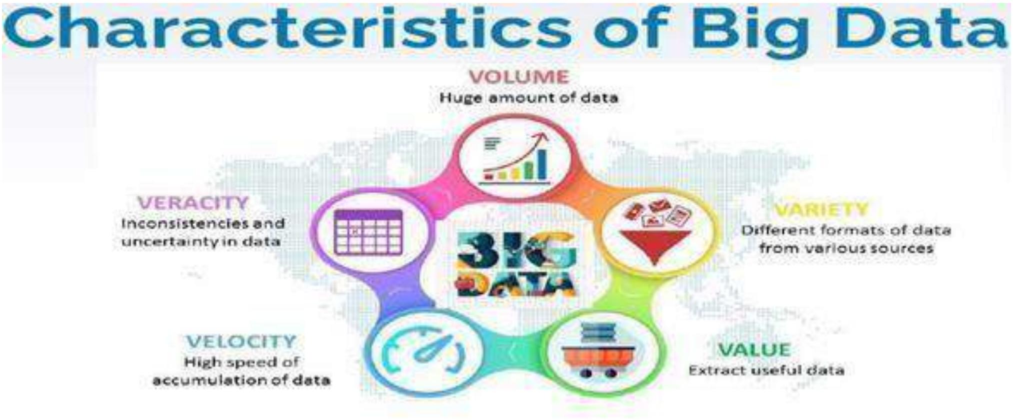
Following are some of the Big Data examples**-**

New York Stock Exchange : The **New York Stock Exchange** is an example of Big Data that generates about **one terabyte** of new trade data per day.

Social Media: The statistic shows that ***500+terabytes*** of new data get ingested into the databases of social media site **Facebook**, every day. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc.

Jet engine :A single **Jet engine** can generate **10+terabytes** of data in **30 minutes** offlight time. With many thousand flights per day, generation of data reaches up to many **Petabytes.**

Big Data Characteristics


Characteristics of Big Data
VOLUME
Huge amount of data
VERACITY
Inconsistencies and uncertainty in data
VARIETY
Different formats of data from various sources
VELOCITY
High speed of accumulation of data
VALUE
Extract useful data
BIG DATA

Volume:

The name Big Data itself is related to an enormous size. Big Data is a vast 'volume' ofdata generated from many sources daily, such as business processes, machines, social media platforms, networks, human interactions, and many more.

Variety:

Big Data can be structured, unstructured, and semi-structured that are being collected from different sources. Data will only be collected from databases and sheets in the past, but these days the data will comes in array forms, that are PDFs, Emails, audios, SM posts, photos, videos, etc.

Veracity :

Veracity means how much the data is reliable. It has many ways to filter or translate the data. Veracity is the process of being able to handle and manage data efficiently. Big Data is also essential in business development.

Value :

Value is an essential characteristic of big data. It is not the data that we process or store. It is **valuable** and **reliable** data that we **store, process**, and also **analyze**.

Velocity :

Velocity plays an important role compared to others. Velocity creates the speed by which the data is created in **real-time**. It contains the linking of incoming **data sets speeds, rate of change**, and **activity bursts**. The primary aspect of Big Data is to provide demanding data rapidly.

Big data velocity deals with the speed at the data flows from sources like **applicationlogs, business processes, networks, and social media sites, sensors, mobile devices,** etc.

Why Big Data?

Big Data initiatives were rated as "extremely important" to 93% of companies. Leveraging a Big Data analytics solution helps organizations to unlock the strategic values and take full advantage of their assets.

It helps organizations:

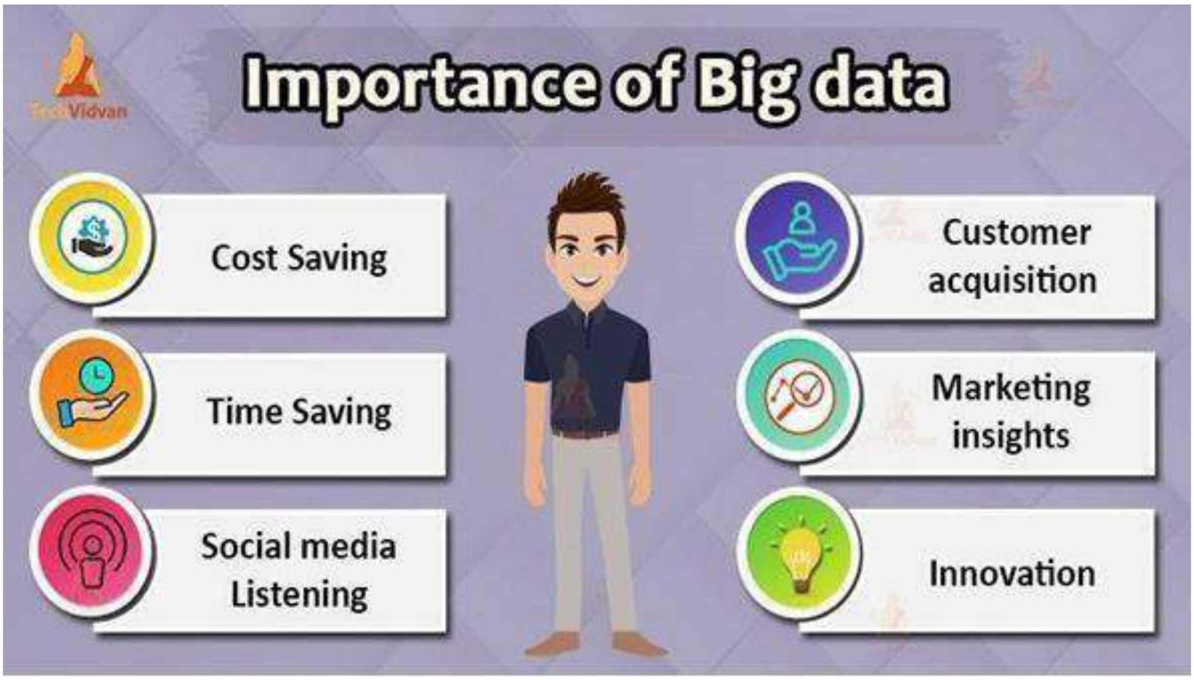
- To understand Where, When and Why their customers buy
- Protect the company's client base with improved loyalty programs
- Seizing cross-selling and upselling opportunities
- Provide targeted promotional information
- Optimize Workforce planning and operations
- Improve inefficiencies in the company's supply chain
- Predict market trends
- Predict future needs
- Make companies more innovative and competitive
- It helps companies to discover new sources of revenue

Companies are using Big Data to know what their customers want, who are their bestcustomers, why people choose different products. The more a company knows aboutits customers, the more competitive it becomes.

We can use it with Machine Learning for creating market strategies based on predictions about customers. Leveraging big data makes companies customer-centric.

Companies can use Historical and real-time data to assess evolving consumers’ preferences. This consequently enables businesses to improve and update their marketing strategies which make companies more responsive to customer needs.

Importance of big data :



Big Data importance doesn’t revolve around the amount of data a company has. Its importance lies in the fact that how the company utilizes the gathered data.
Every company uses its collected data in its own way. More effectively the company uses its data, more rapidly it grows.

The companies in the present market need to collect it and analyze it because:

1. Cost Savings

Big Data tools like Apache Hadoop, Spark, etc. bring cost-saving benefits to businesses when they have to store large amounts of data. These tools help organizations in identifying more effective ways of doing business.

2. **Time-Saving :**

Real-time in-memory analytics helps companies to collect data from various sources. Tools like Hadoop help them to analyze data immediately thus helping in making quick decisions based on the learnings.

3. **Understand the market conditions**

Big Data analysis helps businesses to get a better understanding of market situations .For example, analysis of customer purchasing behavior helps companies to identify the products sold most and thus produces those products accordingly. This helps companies to get ahead of their competitors.

4. **Social Media Listening**

Companies can perform sentiment analysis using Big Data tools. These enable them to get feedback about their company, that is, who is saying what about the company. Companies can use big data tools to improve their online presence.

5. **Boost Customer Acquisition and Retention**

Customers are a vital asset on which any business depends on. No single business can achieve its success without building a robust customer base. But even with a solid customer base, the companies can't ignore the competition in the market.

If we don't know what our customers want then it will degrade companies' success. It will result in the loss of clientele which creates an adverse effect on business growth .Big data analytics helps businesses to identify customer related trends and patterns. Customer behavior analysis leads to a profitable business.

6. **Solve Advertisers Problem and Offer Marketing Insights**

Big data analytics shapes all business operations. It enables companies to fulfill customer expectations. Big data analytics helps in changing the company's product line. It ensures powerful marketing campaigns.

7. The driver of Innovations and Product Development

Big data makes companies capable to innovate and redevelop their products

Challenges of Big Data

When implementing a big data solution, here are some of the common challenges your business might run into, along with solutions.

1. Managing massive amounts of data

It's in the name—big data is big. Most companies are increasing the amount of data they collect daily. Eventually, the storage capacity a traditional data center can provide will be inadequate, which worries many business leaders. Forty-three percent of IT decision-makers in the technology sector worry about this data influx overwhelming their infrastructure [2] .

—

To handle this challenge, companies are migrating their **IT infrastructure** to the cloud. **Cloud storage solutions** can scale dynamically as more storage is needed. **Big data software** is designed to store large volumes of data that can be accessed and queried quickly.

2. Integrating data from multiple sources

The data itself presents another challenge to businesses. There is a lot, but it is also diverse because it can come from a variety of different sources. A business could have analytics data from multiple websites, sharing data from social media, user information from CRM software, email data, and more. None of this data is structured the same but may have to be integrated and reconciled to gather necessary insights and create reports.

To deal with this challenge, businesses use **data integration software**, **ETL** **software**, and **business intelligence software** to map disparate data sources into a common structure and combine them so they can generate accurate reports.

3. Ensuring data quality

Analytics and **machine learning processes** that depend on big data to run also depend on clean, accurate data to generate valid insights and predictions. If the data is corrupted or incomplete, the results may not be what you expect. But as the sources, types, and quantity of data increase, it can be hard to determine if the data has the quality you need for accurate insights.

Fortunately, there are solutions for this. **Data governance applications** will help organize, manage, and secure the data you use in your big data projects while also validating data sources against what you expect them to be and cleaning up corrupted and incomplete data sets. **Data quality software** can also be used specifically for the task of validating and cleaning your data before it is processed.

4. Keeping data secure

Many companies handle data that is sensitive, such as:

- Company data that competitors could use to take a bigger market share of the industry
- Financial data that could give hackers access to accounts
- Personal user information of customers that could be used for identity theft

If a business handles sensitive data, it will become a target of hackers. To protect this data from attack, businesses often hire cybersecurity professionals who keep up to date on security best practices and techniques to secure their systems.

Whether you hire a consultant or keep it in-house, you need to ensure that data is encrypted, so the data is useless without an **encryption** key. Add **identity** and access authorization control to all resources so only the intended users can access it. Implement **endpoint protection software** so malware can't infect the system and real-time **monitoring** to stop threats immediately if they are detected.

5. Selecting the right big data tools

Fortunately, when a business decides to start working with data, there is no shortage of tools to help them do it. At the same time, the wealth of options is also a challenge. **Big data software** comes in many varieties, and their capabilities often overlap. How do you make sure you are choosing the right big data tools? Often, the best option is to hire a consultant who can determine which tools will fit best with what your business wants to do with big data. A big data professional can look at your current and future needs and choose an enterprise data streaming or ETL solution that will collect data from all your data sources and aggregate it. They can configure your cloud services and scale dynamically based on workloads. Once your system is set up with big data tools that fit your needs ,the system will run seamlessly with very little maintenance.

Thinking about hiring a data analytics company to help your business implement a big data strategy? Browse our **list of top data analytics companies**, and learn more about their services in our **hiring guide**.

6. Scaling systems and costs efficiently

If you start building a big data solution without a well-thought-out plan, you can spend a lot of money storing and processing data that is either useless or not exactly what your business needs. Big data is big, but it doesn't mean you have to process all of your data.

When your business begins a data project, start with goals in mind and strategies for how you will use the data you have available to reach those goals. The team involved in implementing a solution needs to plan the type of data they need and the schemas they will use before they start building the system so the project doesn't go in the wrong direction. They also need to create policies for purging old data from the system once it is no longer useful.

7. Lack of skilled data professionals

One of the big data problems that many companies run into is that their current staff have never worked with big data before, and this is not the type of skill set you build overnight. Working with untrained personnel can result in dead ends, disruptions of workflow, and errors in processing.

There are a few ways to solve this problem. One is to **hire a big data specialist** and have that specialist manage and train your data team until they are up to speed. The specialist can either be hired on as a full-time employee or as a consultant who trains your team and moves on, depending on your budget.
Another option, if you have time to prepare ahead, is to offer training to your current team members so they will have the skills once your big data project is in motion.

A third option is to choose one of the **self-service analytics** or **business intelligence** solutions that are designed to be used by professionals who don't have a data science background.

8. Organizational resistance

Another way people can be a challenge to a data project is when they resist change. The bigger an organization is, the more resistant it is to change. Leaders may not see the value in big data, **analytics**, or **machine learning**. Or they may simply not want to spend the time and money on a new project.

This can be a hard challenge to tackle, but it can be done. You can start with a smaller project and a small team and let the results of that project prove the value of big data to other leaders and gradually become a data-driven business. Another option is placing big data experts in leadership roles so they can guide your business towards transformation.

What is Business Intelligence?

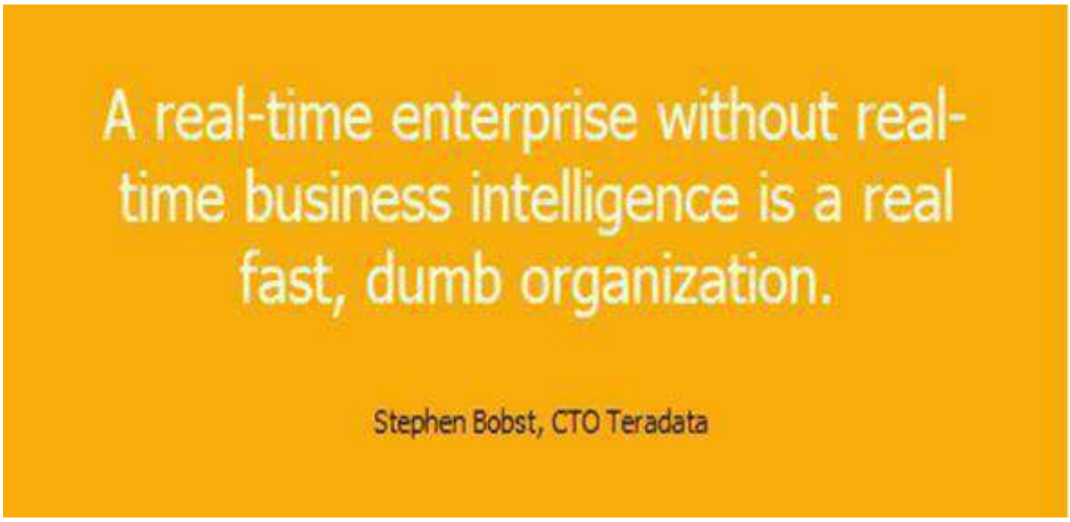
BI(Business Intelligence) is a set of processes, architectures, and technologies that convert raw data into meaningful information that drives profitable business actions. Itis a suite of software and services to transform data into actionable intelligence and knowledge.

BI has a direct impact on organization’s strategic, tactical and operational business decisions.

BI supports fact-based decision making using historical data rather than assumptions and gut feeling.

BI tools perform data analysis and create reports, summaries, dashboards, maps, graphs, and charts to provide users with detailed intelligence about the nature of the business.

Why is BI important?



A real-time enterprise without real-time business intelligence is a real fast, dumb organization.

Stephen Bobst, CTO Teradata

Measurement: creating KPI (Key Performance Indicators) based on historic data

- Identify and set benchmarks for varied processes.
- With BI systems organizations can identify market trends and spot business problems that need to be addressed.
- BI helps on data visualization that enhances the data quality and thereby the quality of decision making.
- BI systems can be used not just by enterprises but SME (Small and Medium Enterprises)

How Business Intelligence systems are implemented?

Here are the steps:

Step 1) Raw Data from corporate databases is extracted. The data could be spread across multiple systems heterogeneous systems.

Step 2) The data is cleaned and transformed into the data warehouse. The table can be linked, and data cubes are formed.

Step 3) Using BI system the user can ask quires, request ad-hoc reports or conduct any other analysis.

Advantages of Business Intelligence

Here are some of the advantages of using Business Intelligence System:

1. **Boost productivity**

With a BI program, It is possible for businesses to create reports with a single click thus saves lots of time and resources. It also allows employees to be more productive on their tasks.

2. **To improve visibility**

BI also helps to improve the visibility of these processes and make it possible to identify any areas which need attention.

3. **Fix Accountability**

BI system assigns accountability in the organization as there must be someone who should own accountability and ownership for the organization's performance against its set goals.

4. **It gives a bird's eye view:**

BI system also helps organizations as decision makers get an overall bird's eye view through typical BI features like dashboards and scorecards.

5. **It streamlines business processes:**

BI takes out all complexity associated with business processes. It also automates analytics by offering predictive analysis, computer modeling, benchmarking and othermethodologies.

6. It allows for easy analytics.

BI software has democratized its usage, allowing even nontechnical or non-analysts users to collect and process data quickly. This also allows putting the power of analytics from the hand's many people.

BI System Disadvantages

1. Cost:

Business intelligence can prove costly for small as well as for medium-sized enterprises. The use of such type of system may be expensive for routine business transactions.

2. Complexity:

Another drawback of BI is its complexity in implementation of data warehouse. It can be so complex that it can make business techniques rigid to deal with.

3. Limited use

Like all improved technologies, BI was first established keeping in consideration the buying competence of rich firms. Therefore, BI system is yet not affordable for many small and medium size companies.

4. Time Consuming Implementation

It takes almost one and half year for data warehousing system to be completely implemented. Therefore, it is a time-consuming process.

UNIT-II

Syllabus

Big data technologies and Databases: Hadoop – Requirement of Hadoop Framework - Design principle of Hadoop –Comparison with other system SQL and RDBMS- Hadoop Components – Architecture -Hadoop 1 vs Hadoop 2.

-

Requirement of Hadoop Framework

- Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models.

- The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers.

- Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Design Principles of Hadoop :

The Design principles of Hadoop on which it works:

a) System shall manage and heal itself

- Automatically and transparently route around failure (Fault Tolerant)
- Speculatively execute redundant tasks if certain nodes are detected to be slow

b) Performance shall scale linearly

- Proportional change in capacity with resource change (Scalability)

C) Computation should move to data

- Lower latency, lower bandwidth (Data Locality)

Simple core, modular and extensible (Economical)

Comparison with other system like SQL

Parameter	Hadoop	SQL
Architecture	Hadoop supports an open-source framework. In Hadoop data sets are distributed across computer/server clusters with parallel data processing features.	SQL stands for **Structured Query Language**. It is based on domain-specific language, used to handle database management operations in relational databases.
Operations	Hadoop is used for storing, processing, retrieving, and pattern extraction from data across a wide range of formats like **XML, Text,** **JSON,** etc.	SQL is used to store, process, retrieve, and pattern mine data stored in a relational database only.
Data Type/ Data update	Hadoop handles both **structured and unstructured data** formats. For data update, Hadoop writes data once but reads data multiple times.	SQL works only for **structured data** but unlike Hadoop, data can be written and read multiple times.
Data Volume Processed	Hadoop is developed for Big Data hence, it usually handles data volumes up to **Terabytes** and **Petabytes**.	SQL works better on low volumes of data, usually in **Gigabytes**.
Data Storage	Hadoop stores data in the form of **key-value pairs**, hash, maps, tables, etc in distributed systems with dynamic schemas.	SQL stores structured data in a **tabular** format using tables only with fixed schemas.
Schema Structure	Hadoop supports **dynamic** schema structure.	SQL supports **static** schema structure.

Data Structures Supported	Hadoop supports **NoSQL** data type structures, columnar data structures, etc. meaning you will have to provide codes for implementation or for rolling back during a transaction.	SQL works on the property of Atomicity, Consistency, Isolation, and Durability **(ACID)** which is fundamental to RDBMS.
Fault Tolerance	Hadoop is **highly** fault-tolerant.	SQL has **good** fault tolerance.
Availability	As Hadoop uses the notion of **distributed computing** and the principle of map-reduce therefore it handles data availability on multiple systems across multiple geo-locations.	SQL supporting databases are usually available **on-prises** or on the cloud, therefore it can't utilize the benefits of distributed computing.
Integrity	Hadoop has **low** integrity.	SQL has **high** integrity.
Scaling	Scaling in Hadoop based system requires connecting computers over the network. Horizontal Scaling with Hadoop is **cheap** and flexible.	Scaling in SQL required purchasing additional SQL servers and configuration which is expensive and time-consuming.
Data Processing	Hadoop supports large-scale batch data processing known as Online Analytical Processing (OLAP).	SQL supports real-time data processing known as Online Transaction Processing (OLTP) thereby making it interactive and batch-oriented.
Execution Time	Statements in Hadoop are executed very **quickly** even when millions of queries are executed at once.	SQL syntax can be **slow** when executed in millions of rows.
Interaction	Hadoop uses appropriate Java Database Connectivity (JDBC) to interact with SQL systems to transfer and receive data between them.	SQL systems can read and write data to Hadoop systems.

Support for ML and AI	Hadoop **supports** advanced machine learning and artificial intelligence techniques.	SQL’s support for ML and AIis **limited** compared to Hadoop.
Skill Level	Hadoop requires an **advanced skill level** for you to be proficient in using it and trying to learn Hadoop as a beginner can be moderately difficult as it requires certain kinds of skill sets.	The SQL skill level required to use it is **intermediate** as it can be learned easily for beginners and entry-level professionals.
Language Supported	Hadoop framework is built with Java programming language.	SQL is a traditional database language used to perform database management operations on relational databases such as MySQL, Oracle, SQL Server, etc.
Use Case	When you need to manage unstructured data, structured data, or semi-structured data in huge volume, Hadoop is a good fit.	SQL performs well in a moderate volume of data and it supports structured data only.
Hardware Configuration	In Hadoop, commodity hardware installation is required on the server.	With SQL supported system, propriety hardware installation is required.
Pricing	Hadoop is a **free** open-source framework.	SQL supporting systems are mostly **licensed**.

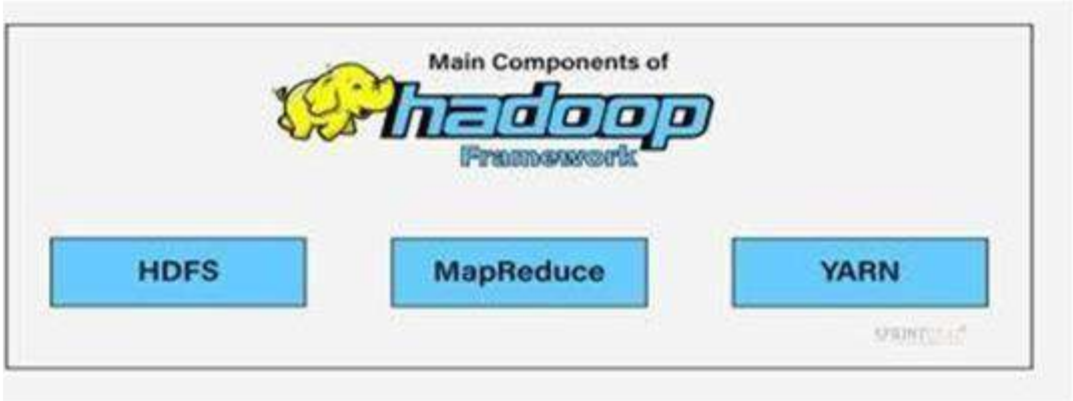
Comparison with other system like RDBMS

Below is the comparison table between Hadoop vs RDBMS.

Feature	RDBMS	Hadoop
Data Variety	Mainly for Structured data	Used for Structured, Semi-Structured, and Unstructured data
Data Storage	Average size data (GBS)	Use for large data sets (Tbs and Pbs)
Querying	SQL Language	HQL (Hive Query Language)
Schema	Required on write (static schema)	Required on reading (dynamic schema)
Speed	Reads are fast	Both reads and writes are fast
Cost	License	Free

Use Case	OLTP (Online transaction processing)	Analytics (Audio, video, logs, etc.), Data Discovery
Data Objects	Works on Relational Tables	Works on Key/Value Pair
Throughput	Low	High
Scalability	Vertical	Horizontal
Hardware Profile	High-End Servers	Commodity/Utility Hardware
Integrity	High (ACID)	Low

Components of Hadoop



There are three **core components of Hadoop** as mentioned earlier. They are HDFS, MapReduce, and YARN. These together form the **Hadoop framework architecture**.

1. HDFS (Hadoop Distributed File System):

It is a data storage system. Since the data sets are huge, it uses a distributed system to store this data. It is stored in blocks where each block is 128 MB. It consists of Name Node and Data Node. There can only be one Name Node but multiple Data Nodes.

Features:

- The storage is distributed to handle a large data pool
- Distribution increases data security
- It is fault-tolerant, other blocks can pick up the failure of one block

3. The **MapReduce framework** is the processing unit. All data is distributed and processed parallelly. There is a Master Node that distributes data amongst

Slave Nodes. The Slave Nodes do the processing and send it back to the Master Node.

Features:

- Consists of two phases, Map Phase and Reduce Phase.
- Processes big data faster with multiples nodes working under one CPU

3. YARN (yet another Resources Negotiator):

It is the resource management unit of the **Hadoop framework**. The data which is stored can be processed with help of YARN using data processing engines like interactive processing. It can be used to fetch any sort of data analysis.

Features:

- It is a filing system that acts as an Operating System for the data stored on HDFS
- It helps to schedule the tasks to avoid overloading any system

Hadoop Architecture

The Hadoop architecture is a package of the file system, MapReduce engine and the HDFS (Hadoop Distributed File System). The MapReduce engine can be MapReduce/MR1 or YARN/MR2.

A Hadoop cluster consists of a single master and multiple slave nodes. The master node includes Job Tracker, Task Tracker, Name Node, and Data Node whereas the slave node includes Data Node and Task Tracker.

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. Itc ontains a master/slave architecture. This architecture consist of a single Name Node performs the role of master, and multiple Data Nodes performs the role of a slave.

Both Name Node and Data Node are capable enough to run on commodity machines. The Java language is used to develop HDFS. So any machine that supports Java language can easily run the Name Node and Data Node software.

Name Node

- It is a single master server exist in the HDFS cluster.
- As it is a single node, it may become the reason of single point failure.
- It manages the file system namespace by executing an operation like the opening, renaming and closing the files.
- It simplifies the architecture of the system.

Data Node

- The HDFS cluster contains multiple Data Nodes.
- Each Data Node contains multiple data blocks.
- These data blocks are used to store data.
- It is the responsibility of Data Node to read and write requests from the file system's clients.
- It performs block creation, deletion, and replication upon instruction from the Name Node.

Job Tracker

- The role of Job Tracker is to accept the MapReduce jobs from client and process the data by using Name Node.
- In response, Name Node provides metadata to Job Tracker.

Task Tracker

- It works as a slave node for Job Tracker.
- It receives task and code from Job Tracker and applies that code on the file .This process can also be called as a Mapper.

MapReduce Layer

The MapReduce comes into existence when the client application submits the MapReduce job to Job Tracker. In response, the Job Tracker sends the request to the appropriate Task Trackers. Sometimes, the Task Tracker fails or time out. In such a case, that part of the job is rescheduled.

Difference between Hadoop 1 and Hadoop 2

Hadoop is an open source software programming framework for storing a large amount of data and performing the computation. Its framework is based on Java programming with some native code in C and shell scripts.

Hadoop 1 vs Hadoop 2

1. Components: In Hadoop 1 we have MapReduce but Hadoop 2 has YARN(Yet A nother Resource Negotiator) and MapReduce version 2.

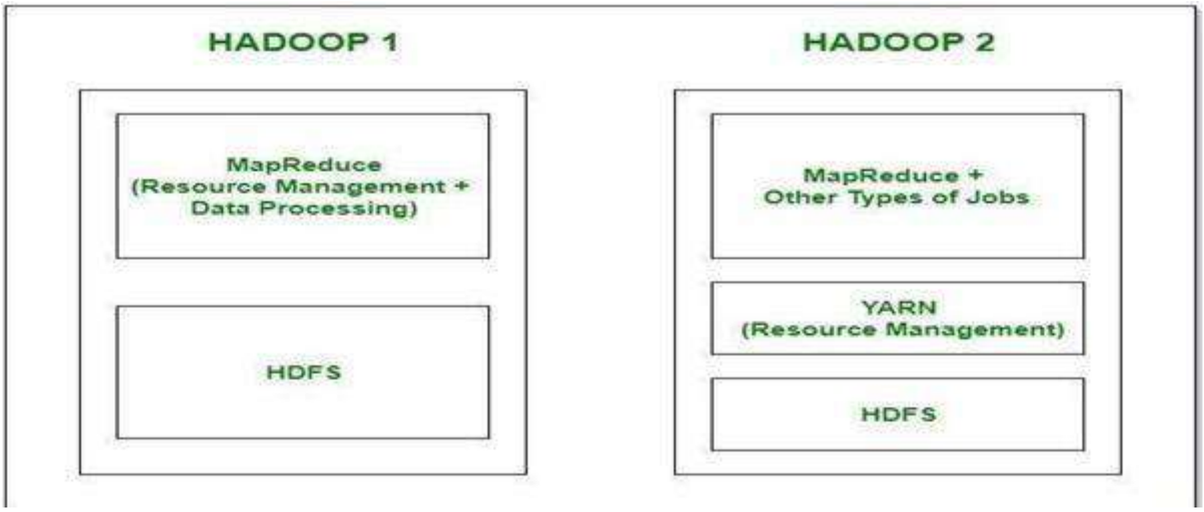
Hadoop 1	**Hadoop 2**
HDFS	HDFS
Map Reduce	YARN / MRv2

4. Daemons:

Hadoop 1	**Hadoop 2**
Namenode	Namenode
Datanode	Datanode
Secondary Namenode	Secondary Namenode
Job Tracker	Resource Manager
Task Tracker	Node Manager

3. Working:

- In *Hadoop 1*, there is HDFS which is used for storage and top of it, Map Reducewhich works as Resource Management as well as Data Processing. Due to thisworkload on Map Reduce, it will affect the performance.
- In *Hadoop 2*, there is again HDFS which is again used for storage and on the topof HDFS, there is YARN which works as Resource Management. It basicallyallocates the resources and keeps all the things going on.

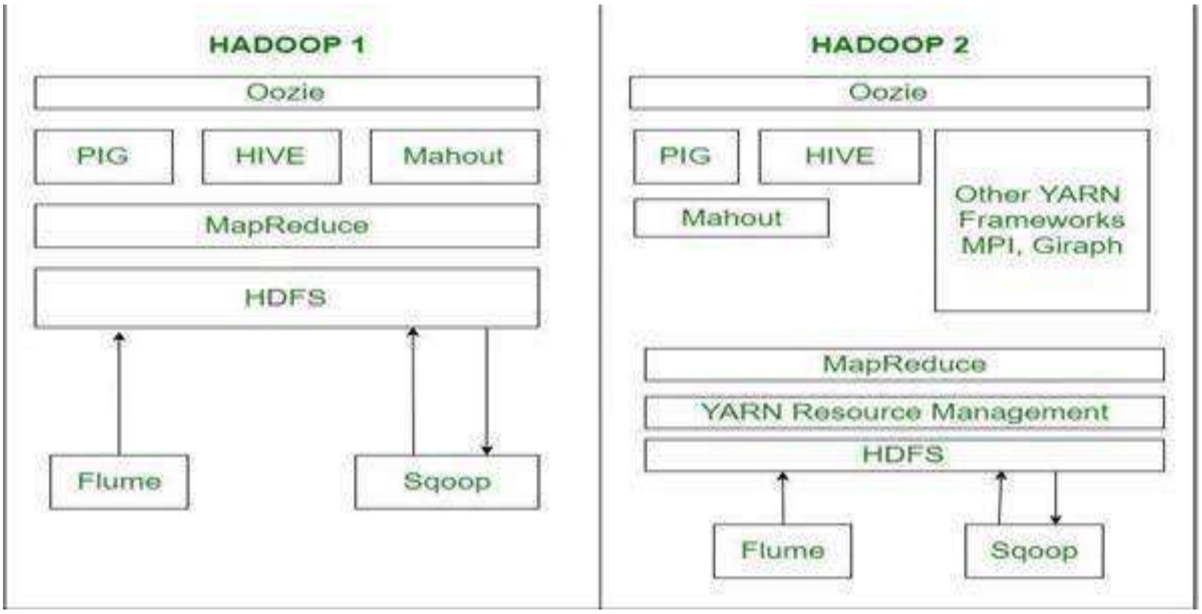


4. Limitations:

Hadoop 1 is a Master-Slave architecture. It consists of a single master and multiple slaves. Suppose if master node got crashed then irrespective of your best slave nodes,your cluster will be destroyed. Again for creating that cluster means copying system files, image files, etc. on another system is too much time consuming which will not betolerated by organizations in today’s time.

Hadoop 2 is also a Master-Slave architecture. But this consists of multiple masters (i.eactive namenodes and standby namenodes) and multiple slaves. If here master nodegot crashed then standby master node will take over it. You can make multiple combinations of active-standby nodes. Thus Hadoop 2 will eliminate the problem of asingle point of failure.

5. Ecosystem



Oozie is basically Work Flow Scheduler. It decides the particular time of jobs to execute according to their dependency.

- Pig, Hive and Mahout are data processing tools that are working on the top ofHadoop.
- Sqoop is used to import and export structured data. You can directly import andexport the data into HDFS using SQL database.
- Flume is used to import and export the unstructured data and streaming data.

UNIT - III

Syllabus

MapReduce and YARN framework: Introduction to MapReduce, Processing data with Hadoop using MapReduce, Introduction to YARN, Architecture, Managing Resources and Applications with Hadoop YARN.

Big data technologies and Databases: NoSQL: Introduction to NoSQL - Features and Types- Advantages & Disadvantages -Application of NoSQL.

Introduction to MapReduce in Hadoop

MapReduce is a software framework and programming model used for processing huge amounts of data. **MapReduce** program work in two phases, namely, Map and Reduce. Map tasks deal with splitting and mapping of data while Reduce tasks shuffle and reduce the data. Hadoop is capable of running MapReduce programs written in various languages: Java, Ruby, Python, and C++. The programs of Map Reduce in cloud computing are parallel in nature, thus are very useful for performing large-scale data analysis using multiple machines in the cluster.

The input to each phase is **key-value** pairs. In addition, every programmer needs to specify two functions: **map function** and **reduce function**.

Processing data with Hadoop using MapReduce

MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment,

MapReduce consists of two distinct tasks – Map and Reduce.

- As the name MapReduce suggests, the reducer phase takes place after the mapper phase has been completed.
- So, the first is the map job, where a block of data is read and processed to produce key-value pairs as intermediate outputs.
- The output of a Mapper or map job (key-value pairs) is input to the Reducer.
- The reducer receives the key-value pair from multiple map jobs.
- Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair)

into a smaller set of tuples or key-value pairs which is the final output.

Let us understand more about MapReduce and its components. MapReduce majorly has the following three Classes. They are,

Mapper Class

The first stage in Data Processing using MapReduce is the **Mapper Class.** Here, RecordReader processes each Input record and generates the respective key-value pair. Hadoop's Mapper store saves this intermediate data into the local disk.

- **Input Split**

It is the logical representation of data. It represents a block of work that contains a single map task in the MapReduce Program.

- **Record Reader**

It interacts with the Input split and converts the obtained data in the form of **Key- Value** Pairs.

Reducer Class

The Intermediate output generated from the mapper is fed to the reducer which processes it and generates the final output which is then saved in the **HDFS.**

Driver Class

The major component in a MapReduce job is a Driver Class. It is responsible for setting up a MapReduce Job to run-in Hadoop. We specify the names of Mapper and Reducer Classes long with data types and their respective job names.

Introduction to YARN

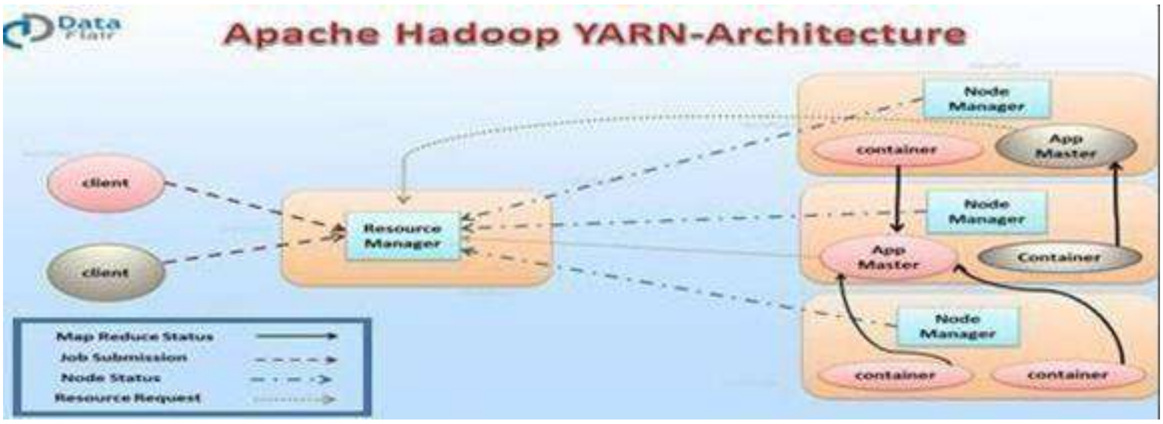
Yet Another Resource Manager takes programming to the next level beyond Java , and makes it interactive to let another application Hbase, Spark etc. to work on it. Different Yarn applications can co-exist on the same cluster so MapReduce, Hbase, Spark all can run at the same time bringing great benefits for manageability and cluster utilization.

Components Of YARN

- **Client:** For submitting MapReduce jobs.
- **Resource Manager:** To manage the use of resources across the cluster
- **Node Manager:** For launching and monitoring the computer containers on machines in the cluster.
- **Map Reduce Application Master:** Checks tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager, and managed by the node managers.

Job tracker & Task tracker were used in previous version of Hadoop, which were responsible for handling resources and checking progress management. However, Hadoop 2.0 has Resource manager and Node Manager to overcome the shortfall of Job tracker & Task tracker.

Hadoop Yarn Architecture



Apache Yarn Framework consists of a master daemon known as "Resource Manager", slave daemon called node manager (one per slave node) and Application Master (oneper application).

1. Resource Manager(RM)

It is the master daemon of Yarn. RM manages the global assignments of resources (CPU and memory) among all the applications. It arbitrates system resources between competing applications. follow Resource Manager guide to learn Yarn Resource manager in great detail.

Resource Manager has two Main components:

- Scheduler
- Application manager

a) Scheduler

The scheduler is responsible for allocating the resources to the running application. The scheduler is pure scheduler it means that it performs no monitoring no tracking for the application and even doesn't guarantees about restarting failed tasks either due to application failure or hardware failures.

b) Application Manager

It manages running Application Masters in the cluster, i.e., it is responsible for starting application masters and for monitoring and restarting them on different nodes in case of failures.

2. Node Manager (NM)

It is the slave daemon of Yarn. NM is responsible for containers monitoring their resource usage and reporting the same to the Resource Manager. Manage the user process on that machine. Yarn Node Manager also tracks the health of the node on which it is running. The design also allows plugging long-running auxiliary services to the NM; these are application-specific services, specified as part of the configuration sand loaded by the NM during startup. A shuffle is a typical auxiliary service by the NMs for MapReduce applications on YARN

3. Application Master(AM)

One application master runs per application. It negotiates resources from the resource manager and works with the node manager. It Manages the application life cycle.

The AM acquires containers from the RM's Scheduler before contacting the corresponding NMs to start the application's individual tasks.

NoSQL

Introduction to NoSQL

We know that MongoDB is a NoSQL Database, so it is very necessary to know about NoSQL Database to understand MongoDB throughly.

What is NoSQL Database

Databases can be divided in 3 types:

1. RDBMS (Relational Database Management System)
2. OLAP (Online Analytical Processing)

3. NoSQL (recently developed database)

NoSQL Database

NoSQL Database is used to refer a non-SQL or non relational database.

It provides a mechanism for storage and retrieval of data other than tabular relations model used in relational databases. NoSQL database doesn't use tables for storing data. It is generally used to store big data and real-time web applications.

Brief History of NoSQL Databases

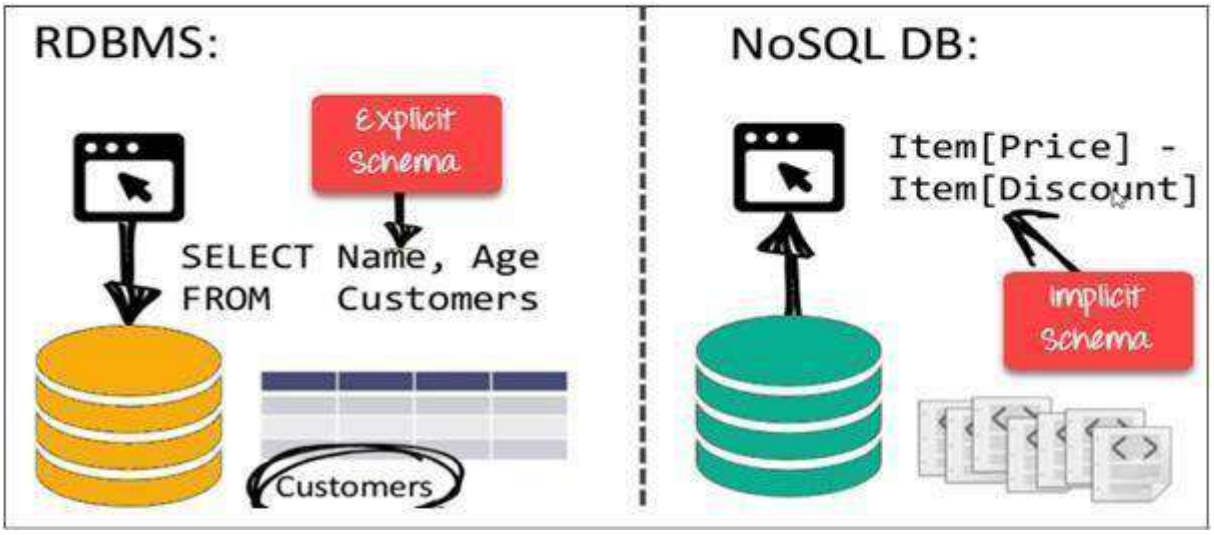
- 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
- 2000- Graph database Neo4j is launched
- 2004- Google Big Table is launched
- 2005- CouchDB is launched
- 2007- The research paper on Amazon Dynamo is released
- 2008- Facebooks open sources the Cassandra project
- 2009- The term NoSQL was reintroduced

Features of NoSQLNonrelational

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn’t require object-relational mapping and data normalization
- No complex features like query languages, query planners, referentialintegrity joins, ACID

Schema-free

- NoSQL databases are either schema-free or have relaxed schemas
- Do not require any sort of definition of the schema of the data
- Offers heterogeneous structures of data in the same domain



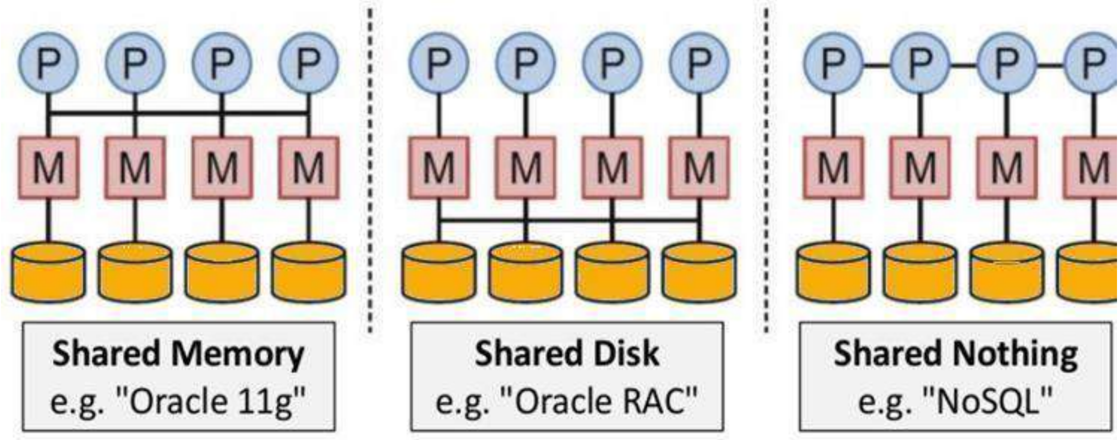
Simple API

- Offers easy to use interfaces for storage and querying data provided
- APIs allow low-level data manipulation & selection methods
- Text-based protocols mostly used with HTTP REST with JSON
- Mostly used no standard based NoSQL query language
- Web-enabled databases running as internet-facing services

Distributed

- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes AsynchronousMulti-Master Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher

distribution.

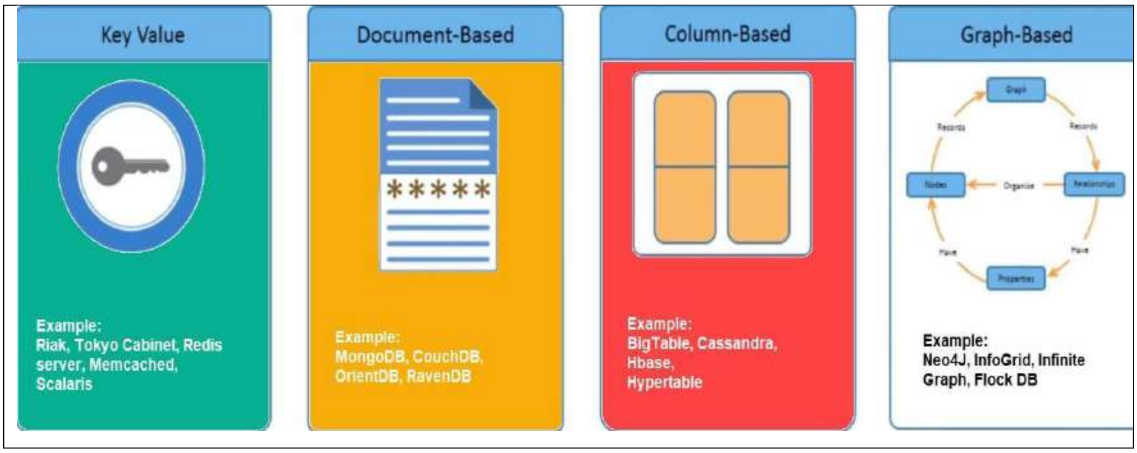


Types of NoSQL Databases

NoSQL Databases are mainly categorized into four types: Key-value pair, Column- oriented, Graph-based and Document-oriented. Every category has its unique attributes and limitations. None of the above-specified database is better to solve all the problems. Users should select the database based on their product needs.

Types of NoSQL Databases:

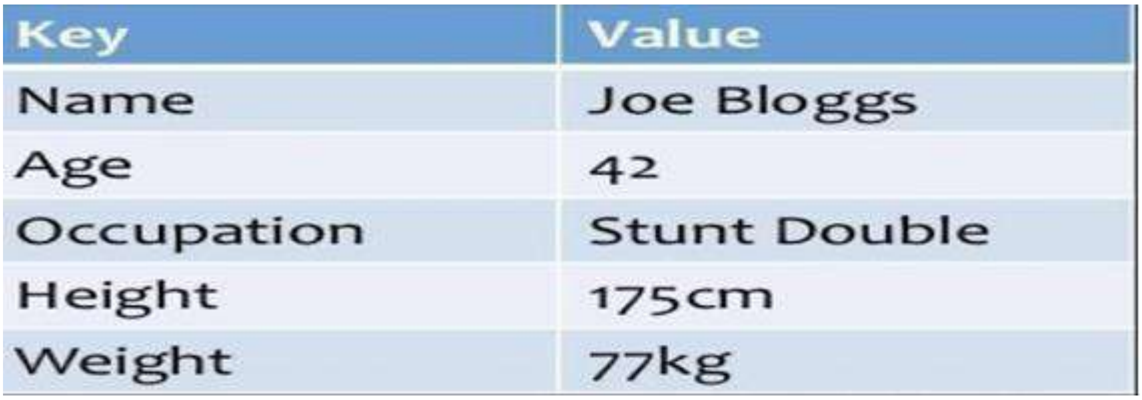
- Key-value Pair Based
- Column-oriented Graph
- Document-oriented
- Graphs based


Key Value
Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris
Document-Based
*****
Example:
MongoDB, CouchDB, OrientDB, RavenDB
Column-Based
Example:
BigTable, Cassandra, Hbase, Hypertable
Graph-Based
Graph
Records
Records
Nodes
Organize
Relationships
Have
Have
Properties
Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Key Value Pair Based

Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load.

Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.



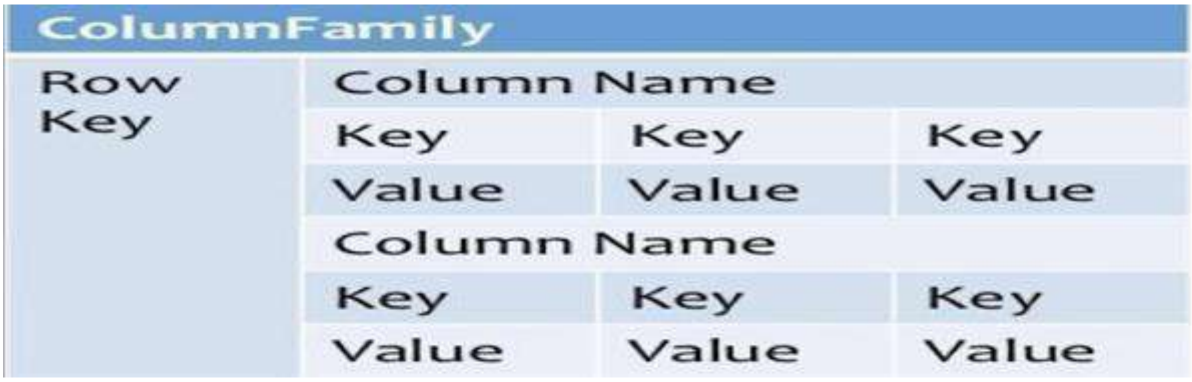
Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

It is one of the most basic NoSQL database example. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.

Redis, Dynamo, Riak are some NoSQL examples of key-value store Data Bases. They are all based on Amazon’s Dynamo paper.

Column-based

Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.



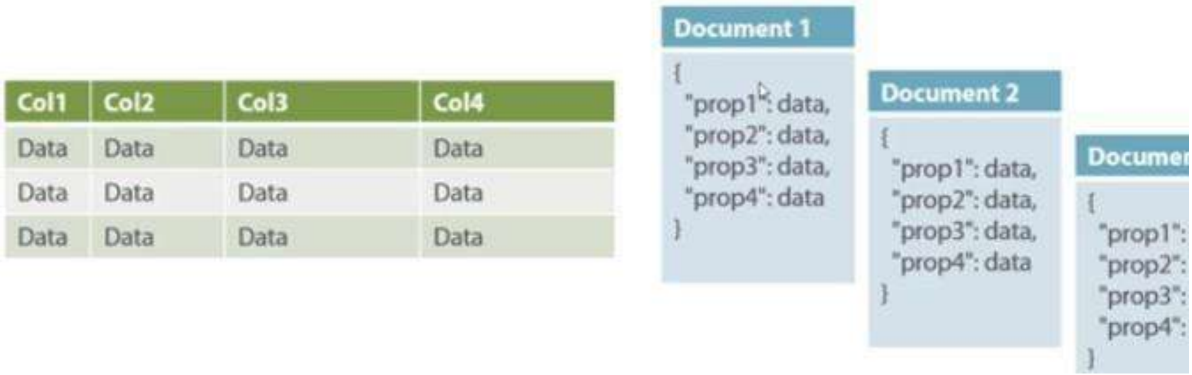
ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.

Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs, HBase, Cassandra, HBase, Hypertable are NoSQL query examples of column based database.

Document-Oriented

Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.



Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

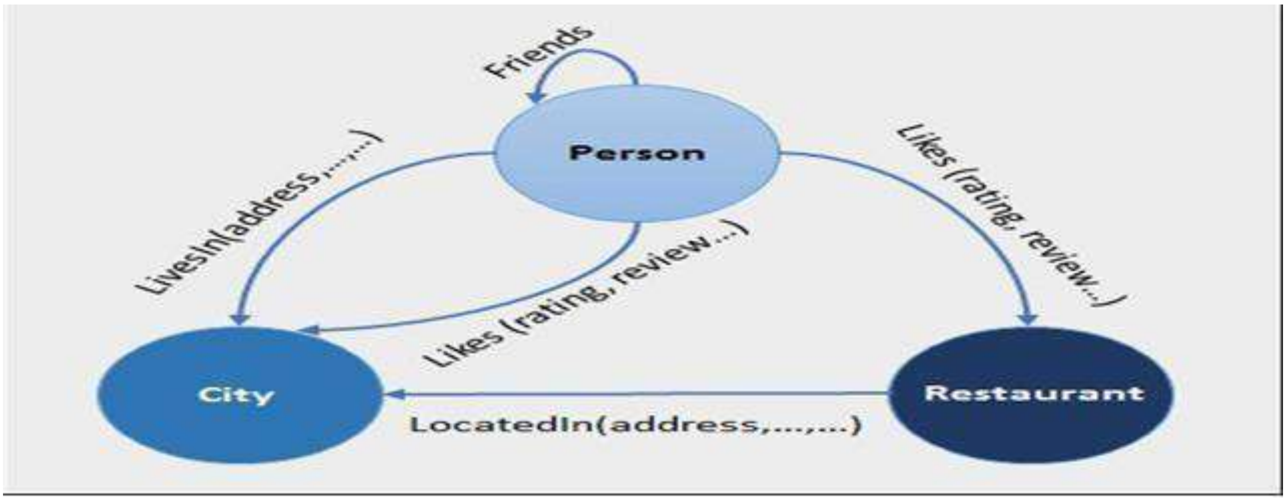
In this diagram on your left you can see we have rows and columns, and in the right, we have a document database which has a similar structure to JSON. Now for the relational database, you have to know what columns you have and so on. However, for a document database, you have data store like JSON object. You do not require todefine which make it flexible.

The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions whichrequire multiple operations or queries against varying aggregate structures.

Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes, MongoDB, are popular Document originated DBMS systems.

Graph-Based

Graph type database stores entities as well the relations amongst those entities. Theentity is stored as a node with the relationship as edges. An edge gives a relationshipbetween nodes. Every node and edge has a unique identifier**.**


Friends
Person
LivesIn(address,......)
Likes (rating, review...)
Likes (rating, review...)
City
Restaurant
LocatedIn(address,...,...)

Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.

Graph base database mostly used for social networks, logistics, spatial data.

Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.

Advantages of NoSQL

- Can be used as Primary or Analytic Data Source
- Big Data Capability
- No Single Point of Failure
- Easy Replication
- No Need for Separate Caching Layer
- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect
- Object-oriented programming which is easy to use and flexible
- NoSQL databases don’t need a dedicated high-performance server
- Support Key Developer Languages and Platforms

- Simple to implement than using RDBMS
- It can serve as the primary data source for online applications.
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Eliminates the need for a specific caching layer to store data
- Offers a flexible schema design which can easily be altered without downtimeor service disruption

Disadvantages of NoSQL

- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency whenmultiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values askeys become difficult
- Doesn’t work as well with relational data
- The learning curve is stiff for new developers
- Open source options so not so popular for enterprises.

Applications of NoSQL Databases

1. **DataMining**

- When it comes to data mining, NoSQL databases are useful in retrieving information for data mining uses. Particularly when it’s about large amounts of data, NoSQL databases store data points in both structured and unstructured formats leading to efficient storage of big data.

- Perhaps when a user wishes to mine a particular dataset from large amounts of data, one can make use of NoSQL databases, to begin with. Data is the building block of technology that has led mankind to such great heights.

- Therefore, one of the most essential fields where NoSQL databases can be put to use is data mining and data storage.

2. Social Media Networking Sites

Social media is full of data, both structured and unstructured. A field that is loaded with tons of data to be discovered, social media is one of the most effective applications of NoSQL databases.

- From comments to posts, user-related information to advertising, social media marketing requires NoSQL databases to be implemented in certain ways to retrieve useful information that can be helpful in certain ways.

- Social media sites like Facebook and Instagram often approach open-source NoSQL databases to extract data that helps them keep track of their users andthe activities going on around their platforms.

3. Software Development

The third application that we will be looking at is software development. Software development requires extensive research on users and the needs of the masses that are met through software development.

- However, a developer must be able to scan through data that is available.

- Perhaps NoSQL databases are always useful in helping software developers keep a tab on their users, their details, and other user-related data that is important to be noted. That said, NoSQL databases are surely helpful in software development.

UNIT - IV

Syllabus

New SQL: Overview of New SQL - Comparing SQL, NoSQL and NewSQL.

Mongo DB: Introduction – Features – Data types – Mongo DB Query language –CRUD operations – Arrays – Functions: Count – Sort – Limit – Skip – Aggregate – MapReduce. Cursors – Indexes – Mongo Import – Mongo Export.

Cassandra: Introduction – Features – Data types – CQLSH – Key spaces – CRUD– operations – Collections – Counter – TTL – Alter commands – Import and Export Querying System tables.

NewSQL

Introduction to NewSQL

NewSQL is a modern relational database system that bridges the gap between SQL and NoSQL. NewSQL databases aim to scale and stay consistent.

NoSQL databases scale while standard SQL databases are consistent. NewSQL attempts to produce both features and find a middle ground. As a result, the databasetype solves the problems in big data fields.

What is NewSQL?

NewSQL is a unique database system that combines ACID compliance with horizontalscaling. The database system strives to keep the best of both worlds. OLTP- based transactions and the high performance of NoSQL combine in a single solution.

Enterprises expect high-quality of data integrity on large data volumes. When either becomes a problem, an enterprise chooses to:

- Improve hardware, or
- Create custom software for distributed databases

Both solutions are expensive on both a software and hardware level. NewSQL strivesto improve these faults by creating consistent databases that scale.

NewSQL Database Features

The main features of NewSQL databases are:

In-memory storage and data processing supply fast query results.

- **Partitioning** scales the database into units. Queries execute on many shardsand combine into a single result.
- **ACID properties** preserve the features of RDBMS.
- **Secondary indexing** results in faster query processing and information retrieval.
- **High availability** due to the database replication mechanism.
- **A built-in crash recovery mechanism** delivers fault tolerance and minimizes downtime.

Difference Between SQL, NoSQL, and NewSQL

The table outlines the main differences between SQL, NoSQL, and NewSQL features.

Feature	**SQL**	**NoSQL**	**NewSQL**
Schema	Relational (table)	Schema-free	Both
SQL	Yes	Depends on the system	Yes,with enhanced features
ACID	Yes	No (BASE)	Yes
OLTP	Partial support	Not supported	Full support
Scaling	Vertical	Horizontal	Horizontal
Distributed	No	Yes	Yes
High availability	Custom	Auto	Built-in
Queries	Low complexity queries	High complexity queries	Both

MongoDB

Introduction to MongoDB

- MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.
- In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.

- MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.
- The manufacturing company 10gen has defined MongoDB as:
- "MongoDB is a scalable, open source, high performance, document-oriented database." - 10gen
- MongoDB was designed to work with commodity servers. Now it is used by the company of all sizes, across all industry.

Features of MongoDB

These are some important features of MongoDB:

1. **Support ad hoc queries**

In MongoDB, you can search by field, range query and it also supports regularexpression searches.

2. **Indexing**

You can index any field in a document.

3. **Replication**

MongoDB supports Master Slave replication.

A master can perform Reads and Writes and a Slave copies data from the master andcan only be used for reads or back up (not writes)

4. **Duplication of data**

MongoDB can run over multiple servers. The data is duplicated to keep the system upand also keep its running condition in case of hardware failure.

5. **Load balancing**

It has an automatic load balancing configuration because of data placed in shards.

6. **Supports map reduce and aggregation tools**.
7. **Uses JavaScript instead of Procedures**.
8. **It is a schema-less database written in C++**.
9. **Provides high performance**.
10. **Stores files of any size easily without complicating your stack**.
11. **Easy to administer in the case of failures**.
12. **It also supports:**
    - JSON data model with dynamic schemas
    - Auto-sharding for horizontal scalability

- Built in replication for high availability
- Now a day many companies using MongoDB to create new types of applications, improve performance and availability.

MongoDB Datatypes

Following is a list of usable data types in MongoDB.

Data Types	Description
String	String is the most commonly used datatype. It is used to store data. Astring must be UTF 8 valid in mongodb.
Integer	Integer is used to store the numeric value. It can be 32 bit or 64 bit depending on the server you are using.
Boolean	This datatype is used to store boolean values. It just shows YES/NOvalues.
Double	Double datatype stores floating point values.
Min/Max Keys	This datatype compare a value against the lowest and highest bson elements.
Arrays	This datatype is used to store a list or multiple values into a single key.
Object	Object datatype is used for embedded documents.
Null	It is used to store null values.
Symbol	It is generally used for languages that use a specific type.
Date	This datatype stores the current date or time in unix time format. It makes you possible to specify your own date time by creating object ofdate and pass the value of date, month, year into it.

Mongo DB Query language -CRUD Operations

CRUD operations describe the conventions of a user-interface that let users view, search, and modify parts of the database.

MongoDB documents are modified by connecting to a server, querying the proper documents, and then changing the setting properties before sending the data back tothe database to be updated. CRUD is data-oriented, and it's standardized according to HTTP action verbs.

When it comes to the individual CRUD operations:

- The Create operation is used to insert new documents in the MongoDB database.
- The Read operation is used to query a document in the database.
- The Update operation is used to modify existing documents in the database.
- The Delete operation is used to remove documents in the database.

How to Perform CRUD Operations

Now that we've defined MongoDB CRUD operations, we can take a look at how to carry out the individual operations and manipulate documents in a MongoDB database. Let's go into the processes of creating, reading, updating, and deleting documents, looking at each operation in turn.

Create Operations

For MongoDB CRUD, if the specified collection doesn't exist, the create operation willcreate the collection when it's executed. Create operations in MongoDB target a singlecollection, not multiple collections. Insert operations in MongoDB are atomic on a single document level.

MongoDB provides two different create operations that you can use to insert documents into a collection:

- db.collection.insertOne()
- db.collection.insertMany()

Read Operations

The read operations allow you to supply special query filters and criteria that let you specify which documents you want. The MongoDB documentation contains more information on the available query filters. Query modifiers may also be used to change how many results are returned.

MongoDB has two methods of reading documents from a collection:

- db.collection.find()
- db.collection.findOne()

Update Operations

Like create operations, update operations operate on a single collection, and they areatomic at a single document level. An update operation takes filters and criteria to select the documents you want to update.

You should be careful when updating documents, as updates are permanent and can'tbe rolled back. This applies to delete operations as well.

For MongoDB CRUD, there are three different methods of updating documents:

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()

Delete Operations

Delete operations operate on a single collection, like update and create operations. Delete operations are also atomic for a single document. You can provide delete operations with filters and criteria in order to specify which documents you would like to delete from a collection. The filter options rely on the same syntax that read operations utilize.

MongoDB has two different methods of deleting records from a collection:

- db.collection.deleteOne()
- db.collection.deleteMany()

Introduction to MongoDB Array

MongoDB Array is a flexible document structure; it will make it possible to have a fieldwith array as a value in MongoDB. This is nothing but a simple list of values, and it can take many forms in MongoDB. We can define an array of string, integer, embedded documents, Jason and BSON data types, array in it can be defined as anyform of data types. If we have made student collection and make the field as grade and grade are divided into three parts, such as MongoDB array, Array is essential anduseful in MongoDB.

Syntax:

{< array field >: {< operator1> : <value1>, < operator2> : <value2>, < operator3> : <value3>, ….. }}

Parameters to MongoDB Array:

1. Array field: Array field is defined as the field name of the collection on which we create or define array values. The array field is significant while defining any array.

2. Operator: The operator is defined as which values we have to create the array. Theoperator name is specific to the value name in the array.

3. Value: Value in the array is defined as the actual value of array on which we have defining array in MongoDB. Value is significant while defining an array.

Examples to Implement MongoDB Array

Example #1

In the below example, we have to define an array of emp_skills after defining we haveinserted an array of documents in emp_count collection:

Code:

```
db.emp_count.find()
```

Output:

Mongodb count() Method

The count() method counts the number of documents that match the selection criteria.It returns the number of documents that match the selection criteria. It takes two arguments first one is the selection criteria and the other is optional.

- This method is equivalent to db.collection.find().count().
- You cannot use this method in transactions.
- One a shared cluster, if you use this method without a query predicate, then it will return an inaccurate count if orphaned documents exist or if a chunk migration is in progress. So, to avoid such type of situation use db.collection.aggregate() method.

Syntax:

db.collection.count(query)

Parameters:

Name	Description	Required / Optional	Type
query	The query selection criteria.	Required	document

The sort() Method

To sort documents in MongoDB, you need to use **sort()** method. The method acceptsa document containing a list of fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descendingorder.

Syntax:

The basic syntax of **sort()** method is as follows −

```
>db.COLLECTION_NAME.find().sort({KEY:1})
```

MongoDB limit() Method

In MongoDB, limit() method is used to limit the fields of document that you want to show. Sometimes, you have a lot of fields in collection of your database and have to retrieve only 1 or 2. In such case, limit() method is used.

The MongoDB limit() method is used with find() method.

Syntax:

db.COLLECTION_NAME.find().limit(NUMBER)

MongoDB Skip() Method

In MongoDB, the skip() method will skip the first n document from the query result, youjust need to pass the number of records/documents to be skipped. It basically removesthe first n documents from the result set. For example, if your result has 5 records in itand you want to remove the first two records from it then you can use skip(2) at the end of the query. Or in other words, this method call on a cursor to control from whereMongoDB starts returning the results.

Syntax :

cursor.skip(<offset>)

The aggregate() Method

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In SQL count(*) and with group by is an equivalent of MongoDB aggregation.

Syntax:

Basic syntax of **aggregate()** method is as follows −

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

MongoDB Map Reduce()

MongoDB mapReduce() method can be used to aggregate documents in a MongoDB Collection.

Now we shall learn to use mapReduce() function for performing aggregation operations on a MongoDB Collection, with the help of examples.

Syntax:

```
> db.collection.mapReduce(
    function() {emit(key,value);}, // map function function(key,values)
    {return reduceFunction},  // reduce function
    { out: collection }
)
```

MongoDB Cursor Methods

The MongoDB cursor methods modifies the way that the specified query is executed.Following are the list of the cursor methods.

1 cursor.addOption(flag)

2. Cursor.batchSize(size)

3. cursor.close()

4. cursor.collation(<collation document>)

5. cursor.forEach(function)

7. cursor.limit()

8. cursor.map(function)

9. cursor.max()

10. cursor.min()

11. cursor.tailable()

12. cursor.toArray()

Indexing in MongoDB

MongoDB uses indexing in order to make the query processing more efficient. If thereis no indexing, then the MongoDB must scan every document in the collection and retrieve only those documents that match the query. Indexes are special data structures that stores some information related to the documents such that it becomeseasy for MongoDB to find the right data file. The indexes are order by the value of thefield specified in the index.

Creating an Index :

MongoDB provides a method called createIndex() that allows user to create an index.

Syntax:

db.COLLECTION_NAME.createIndex({KEY:1})

The key determines the field on the basis of which you want to create an index and 1 (or -1) determines the order in which these indexes will be arranged(ascending or descending).

What is mongoimport

The mongoimport command is used to import your content from an extended JSON, CSV, or TSV export created by mongoexport. It also supports restoring or importing data from other third-party export tools.

This command is of immense help when it comes to managing your MongoDB database. It's **super-fast** and **multi-threaded**, more so than any custom script you might write to do your import operation. The mongoimport command can be combinedwith other MongoDB command-line tools, such as **jq** for JSON manipulation, **csvkit** for CSV manipulation, or even **curl** for dynamically downloadingdata files from servers on the internet.

Syntax:

The mongoimport command has the following syntax:

```
mongoimport <options> <connection-string> <file>
```

What is Mongoexport

mongoexport command to Export Data from a Collection mongoexport command is used to export MongoDB Collection data to a CSV or JSON file. By default, the mongoexport command connects to mongod instance running on the localhost port number 27017.

Syntax:

mongoexport --db DB_NAME --collection COLLECTION_name --fields

Field_name(s) --type=[csv or JSON} --out=Name-Path-Output-File **where,**

DB_NAME – Name of the Database of the Collection to be exported

COLLECTION_name - Name of Collection of DB_NAME to be exported

Field_name(s) - Name of the Field (or multiple fields separated by comma (,)) to be exported. It is optional in case of CSV file. If not specified, all the fields of the collectionwill be exported to JSON file.

Type – CSV or JSON format

Name-Path-Output-File – Name and path of the output (exported) file. It is also optional.

But it is suggested to properly specify the name and path of the exported file.

When exporting to CSV format, must specify the fields in the documents to be exported

otherwise it will display error.

When exporting to JSON format, _id field will also be exported by default. While thesame

will not be exported if exporting as CSV file until not specified in field list.

Cassandra

What is Cassandra

Apache Cassandra is highly scalable, high performance, distributed NoSQL database.Cassandra is designed to handle huge amount of data across many commodity servers, providing high availability without a single point of failure.

Cassandra has a distributed architecture which is capable to handle a huge amount of data. Data is placed on different machines with more than one replication factor to attain a high availability without a single point of failure.

Important Points of Cassandra

- Cassandra is a column-oriented database.
- Cassandra is scalable, consistent, and fault-tolerant.
- Cassandra's distribution design is based on Amazon's Dynamo and its datamodel on Google's Bigtable.

- Cassandra is created at Facebook. It is totally different from relational database management systems.
- Cassandra follows a Dynamo-style replication model with no single point of failure, but adds a more powerful "column family" data model.
- Cassandra is being used by some of the biggest companies like Facebook,Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

Features of Cassandra

There are a lot of outstanding technical features which makes Cassandra very popular.Following is a list of some popular features of Cassandra:

High Scalability

Cassandra is highly scalable which facilitates you to add more hardware to attach more customers and more data as per requirement.

Rigid Architecture

Cassandra has not a single point of failure and it is continuously available for business-critical applications that cannot afford a failure.

Fast Linear-scale Performance

Cassandra is linearly scalable. It increases your throughput because it facilitates you to increase the number of nodes in the cluster. Therefore it maintains a quick responsetime.

Fault tolerant

Cassandra is fault tolerant. Suppose, there are 4 nodes in a cluster, here each node has a copy of same data. If one node is no longer serving then other three nodes canserved as per request.

Flexible Data Storage

Cassandra supports all possible data formats like structured, semi-structured, and unstructured. It facilitates you to make changes to your data structures according to your need.

Easy Data Distribution

Data distribution in Cassandra is very easy because it provides the flexibility to distribute data where you need by replicating data across multiple data centers.

Transaction Support

Cassandra supports properties like Atomicity, Consistency, Isolation, and Durability (ACID).

Fast writes

Cassandra was designed to run on cheap commodity hardware. It performs blazinglyfast writes and can store hundreds of terabytes of data, without sacrificing the read efficiency.

Cassandra Data Types

Cassandra supports different types of data types. Lets see the different data types in the following table:

CQL Type	Constants	Description
ascii	Strings	US-ascii character string
Big int	Integers	64-bit signed long
blob	blobs	Arbitrary bytes in hexadecimal
Boolean	Booleans	True or False
counter	Integers	Distributed counter values 64bit
decimal	Integers, Floats	Variable precision decimal
double	Integers, Floats	64-bit floating point
float	Integers, Floats	32-bit floating point
frozen	Tuples, collections, user defined types	stores cassandra types
inet	Strings	IP address in ipv4 or ipv6 format

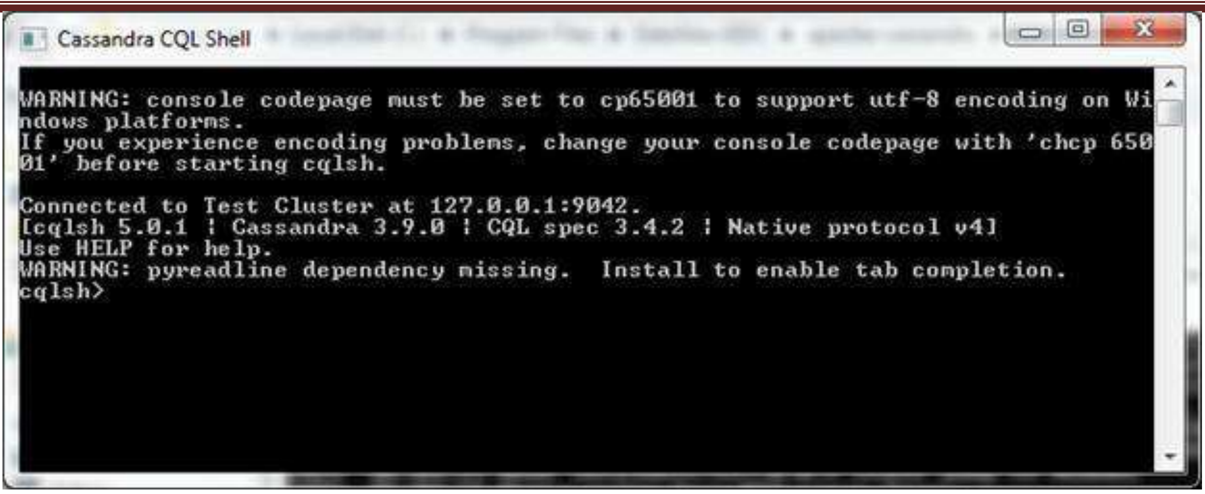
int	Integers	32 bit signed integer
list		Collection of elements
map		JSON style collection of elements
set		Collection of elements
text	strings	UTF-8 encoded strings
timestamp	Integers, Strings	ID generated with date plus time
timeuuid	uuids	Type 1 uuid
tuple		A group of 2,3 fields
uuid	uuids	Standard uuid
varchar	strings	UTF-8 encoded string
varint	Integers	Arbitrary precision integer

Cassandra CQLsh

Cassandra CQLsh stands for Cassandra CQL shell. CQLsh specifies how to use Cassandra commands. After installation, Cassandra provides a prompt Cassandra query language shell (cqlsh). It facilitates users to communicate with it.

Cassandra commands are executed on CQLsh. It looks like this:

Start CQLsh:



```
WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Wi
ndows platforms.
If you experience encoding problems, change your console codepage with 'chcp 650
01' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.9.0 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing.  Install to enable tab completion.
cqlsh>
```

CQLsh provides a lot of options which you can see in the following table:

Options	**Usage**
help	This command is used to show help topics about the options ofCQLsh commands.
version	it is used to see the version of the CQLsh you are using.
color	it is used for colored output.
debug	It shows additional debugging information.
execute	It is used to direct the shell to accept and execute a CQL command.
file= "file name"	By using this option, cassandra executes the command in the givenfile and exits.
no-color	It directs cassandra not to use colored output.
u "username"	Using this option, you can authenticate a user. The default username is: cassandra.
p "password"	Using this option, you can authenticate a user with a password. Thedefault password is: cassandra.

What is Keyspace?

A keyspace is an object that is used to hold column families, user defined types. A keyspace is like RDBMS database which contains column families, indexes, user defined types, data center awareness, strategy used in keyspace, replication factor, etc.

In Cassandra, "Create Keyspace" command is used to create keyspace.

Syntax:

CREATE KEYSPACE <identifier> WITH <properties>

Or

Create keyspace KeyspaceName with replicaton={'class':strategy name, 'replication_factor': No of replications on different nodes}

Create keyspace KeyspaceName with replicaton={'class':strategy name, 'replication_factor': No of replications on different nodes}

CRUD Operations in Mongodb

1. Creating Data in a Table

You can insert data into the columns of a row in a table using the command **INSERT**. Given below is the syntax for creating data in a table.

```
INSERT INTO <tablename>
(<column1 name>, <column2 name>... )
VALUES (<value1>, <value2>....)
USING <option>
```

2. Updating Data in a Table

UPDATE is the command used to update data in a table. The following keywords areused while updating data in a table −

- **Where** − This clause is used to select the row to be updated.

- **Set** − Set the value using this keyword.
- **Must** − Includes all the columns composing the primary key.

While updating rows, if a given row is unavailable, then UPDATE creates a fresh row.Given below is the syntax of UPDATE command −

```
UPDATE <tablename>
SET <column name> = <new value>
<column name> = <value>.....WHERE
<condition>
```

3. Reading Data using Select Clause

SELECT clause is used to read data from a table in Cassandra. Using this clause, youcan read a whole table, a single column, or a particular cell. Given below is the syntax of SELECT clause.

```
SELECT FROM <tablename>
```

4. Deleting Datafrom a Table

You can delete data from a table using the command **DELETE**. Its syntax is as follows –

```
DELETE FROM <identifier> WHERE <condition>;
```

Cassandra Collections

Cassandra collections are used to handle tasks. You can store multiple elements in collection. There are three types of collection supported by Cassandra:

- Set
- List
- Map

Cassandra Counter

The counter is a special column used to store a number that this changed increments. For example, you might use a counter column to count the number of times a page isviewed. So, we can define a counter in a dedicated table only and use that counter datatype.

Restriction on the counter column:

1. Counter column cannot index, delete or re-add a counter column.
2. All non-counter columns in the table must be defined as a part of the primarykey.
3. To load data in a counter column or to increase or decrease the value of thecounter, use the update command.

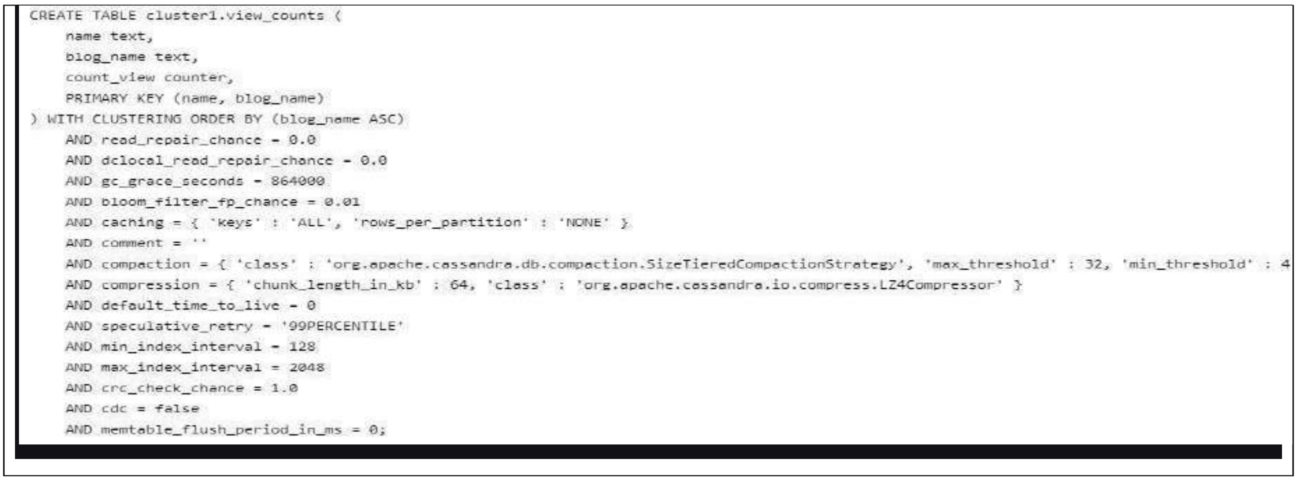
Now, we are going to create table with a Counter column. let’s have a look.

```
Create table View_Counts(
  count_view counter,
  name varchar,
  blog_name text,
  primary key(name, blog_name)
 );
```

Let’s see the table schema.

describe table View_Counts;

Output:



```
CREATE TABLE cluster1.view_counts (
    name text,
    blog_name text,
    count_view counter,
    PRIMARY KEY (name, blog_name)
) WITH CLUSTERING ORDER BY (blog_name ASC)
    AND read_repair_chance = 0.0
    AND dclocal_read_repair_chance = 0.0
    AND gc_grace_seconds = 864000
    AND bloom_filter_fp_chance = 0.01
    AND caching = { 'keys' : 'ALL', 'rows_per_partition' : 'NONE' }
    AND comment = ''
    AND compaction = { 'class' : 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold' : 32, 'min_threshold' : 4
    AND compression = { 'chunk_length_in_kb' : 64, 'class' : 'org.apache.cassandra.io.compress.LZ4Compressor' }
    AND default_time_to_live = 0
    AND speculative_retry = '99PERCENTILE'
    AND min_index_interval = 128
    AND max_index_interval = 2048
    AND crc_check_chance = 1.0
    AND cdc = false
    AND memtable_flush_period_in_ms = 0;
```

TTL command in Cassandra

Time to Live (TTL) is play an important role while if we want to set the time limit of a column and we want to automatically delete after a point of time then at the time usingTTL keyword is very useful to define the time limit for a particular column.

1. In Cassandra Both the INSERT and UPDATE commands support setting a timefor data in a column to expire.
2. It is used to set the time limit for a specific period of time. By USING TTL clausewe can set the TTL value at the time of insertion.
3. We can use TTL function to get the time remaining for a specific selected query.
4. At the point of insertion, we can set expire limit of inserted data by using TTL clause. Let us consider if we want to set the expire limit to two days then we need to define its TTL value.
5. By using TTL we can set the expiration period to two days and the value of TTLwill be 172800 seconds. Let's understand with an example.

Table : student_Registration

To create the table used the following CQL query.

```
CREATE TABLE student_Registration(Id int PRIMARY KEY,Name text,Event text);
```

Insertion using TTL :

To insert data by using TTL then used the following CQL query.

```
INSERT INTO student_Registration (Id, Name, Event) VALUES
    (101, 'Ashish', 'Ninza') USING TTL 172800;
INSERT INTO student_Registration (Id, Name, Event) VALUES
    (102, 'Ashish', 'Code') USING TTL 172800;
INSERT INTO student_Registration (Id, Name, Event) VALUES
    (103, 'Aksh', 'Ninza') USING TTL 172800;
```

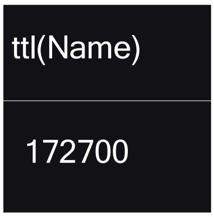
Output:

Id	Name	Event
101	Ashish	Ninza
102	Ashish	Code
103	Aksh	Ninza

Now, to determine the remaining time to expire for a specific column used the followingCQL query.

```
SELECT TTL (Name)
from student_Registration
WHERE Id = 101;
```

Output:

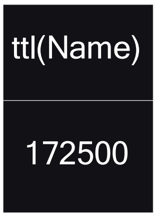


```
ttl(Name)
172700
```

It will decrease as you will check again for its TTL value just because of TTL time limit.Now, used the following CQL query to check again.

```
SELECT TTL (Name)
from student_Registration
WHERE Id = 101;
```

Output:



```
ttl(Name)
172500
```

Alter command in Cassandra

You can alter a table using the command ALTER TABLE. Given below is the syntax for creating a table.

Syntax:

```
ALTER (TABLE | COLUMNFAMILY) <tablename> <instruction>
```

Using ALTER command, you can perform the following operations −

- Add a column
- Drop a column

Adding a Column

Using ALTER command, you can add a column to a table. While adding columns, youhave to take care that the column name is not conflicting with the existing column names and that the table is not defined with compact storage option.

Given below is the syntax to add a column to a table.

```
ALTER TABLE table name
ADD  new column datatype;
```

Dropping a Column

Using ALTER command, you can delete a column from a table. Before dropping a column from a table, check that the table is not defined with compact storage option. Given below is the syntax to delete a column from a table using ALTER command.

```
ALTER table name
DROP column name;
```

Data Import and Export

To import or export data from or into Cassandra, CQL provides a simple statement called "COPY" that can be used to either import or export data. The COPY statementcan either export data from a table into a CSV file or import data from a CSV file and into a table. The "COPY" command can be used with the below syntax:

To import data:

```
COPY table_name ( column, ...)
FROM ( 'file_name' | STDIN )
WITH option = 'value' AND ...
```

To Export data:

```
COPY table_name ( column , ... )
TO ( 'file_name' | STDOUT ) WITH
option = 'value' AND ...
```

Querying a system table

The system keyspace includes a number of tables that contain details about your Cassandra database objects and cluster configuration.

Cassandra populates these tables and others in the system keyspace.

Table name	**Column name**	**Comment**
schema_keyspaces	keyspace_name, durable_writes, strategy_class, strategy_options	None
local	"key", bootstrapped, cluster_name, cql_version, data_center, gossip_generation, native_protocol_version,partitioner, rack, release_version, ring_id,	nformation a node has about itself and asuperset of gossip.

	schema_version, thrift_version, tokens set, truncated at map	
peers	peer, data_center, rack, release_version, ring_id, rpc_address, schema_version, tokens	Each node records what other nodes tell it about themselves over the gossip.
schema_columns	keyspace_name, columnfamily_name, column_name, component_index, index_name, index_options, index_type, validator	Used internally with compound primarykeys.
schema_column families	See comment.	Inspect schema_column families to get detailed information about specific tables.

UNIT – V

Syllabus

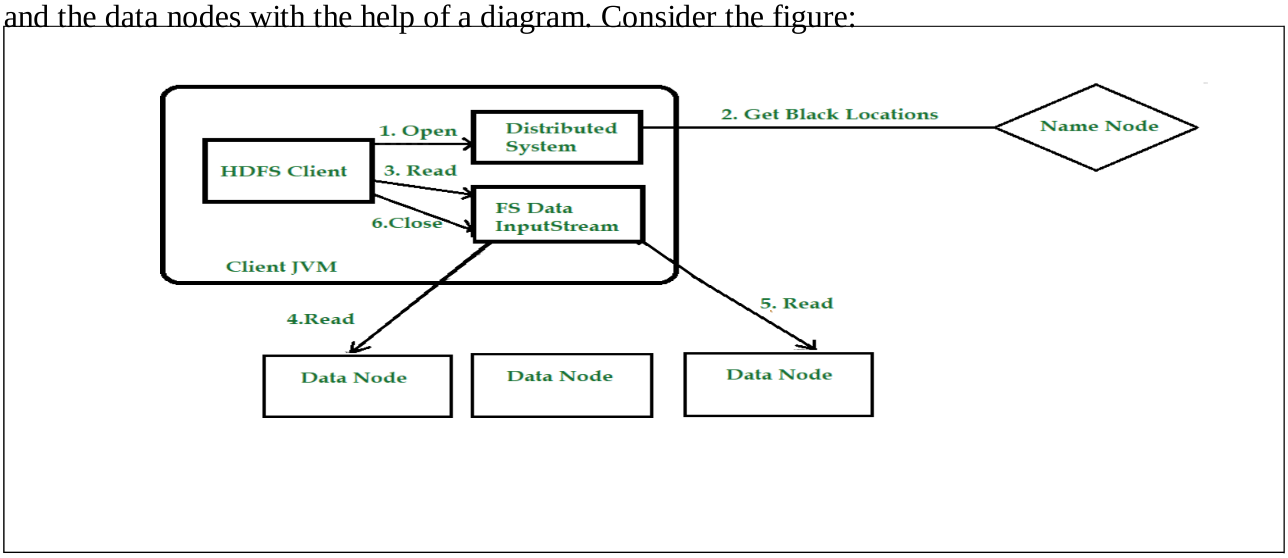
(Big Data Frame Works for Analytics) Hadoop Frame Work: Map Reduce Programming: I/O formats, Map side join-Reduce Side Join-Secondary Sorting- Pipelining MapReduce jobs **Spark Frame Work:** Introduction to Apache spark-How spark works, Programming with RDDs: Create RDDspark Operations-Data Frame.

Map Reduce Programming

I/O formats

Anatomy of File Read in HDFS:

Let's get an idea of how data flows between the client interacting with HDFS, the namenode, and the data nodes with the help of a diagram. Consider the figure:



Step 1: The client opens the file it wishes to read by calling open() on the File System Object(which for HDFS is an instance of Distributed File System).

Step 2: Distributed File System(DFS) calls the name node, using remote procedure calls (RPCs), to determine the locations of the first few blocks in the file. For each block, the name node returns the addresses of the data nodes that have a copy of thatblock. The DFS returns an FSDataInputStream to the client for it to read data from.

FSDataInputStream in turn wraps a DFSInputStream, which manages the data node and name node I/O.

Step 3: The client then calls read() on the stream. DFSInputStream, which has storedthe info node addresses for the primary few blocks within the file, then connects to theprimary (closest) data node for the primary block in the file.

Step 4: Data is streamed from the data node back to the client, which calls read() repeatedly on the stream.

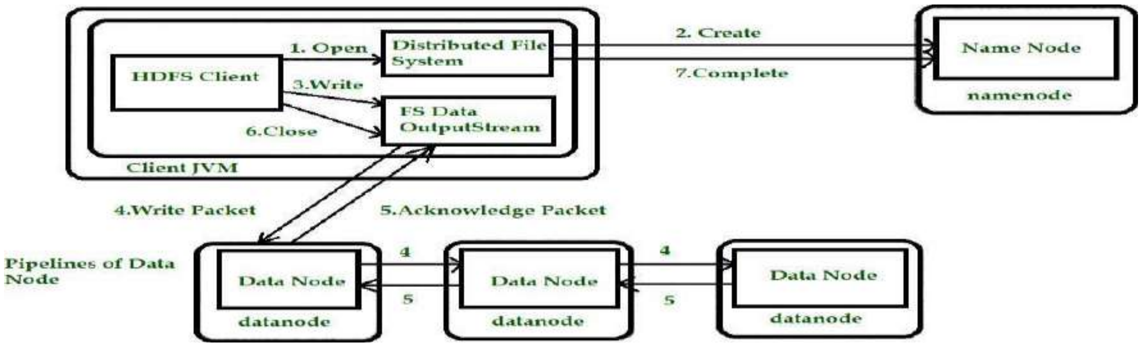
Step 5: When the end of the block is reached, DFSInputStream will close the connection to the data node, then finds the best data node for the next block. This happens transparently to the client, which from its point of view is simply reading an endless stream. Blocks are read as, with the DFSInputStream opening new connections to data nodes because the client reads through the stream. It will also callthe name node to retrieve the data node locations for the next batch of blocks as needed.

Step 6: When the client has finished reading the file, a function is called, close() on the FSDataInputStream.

Anatomy of File Write in HDFS:

Next, we'll check out how files are written to HDFS. Consider figure 1.2 to get a better understanding of the concept.

Note: HDFS follows the Write once Read many times model. In HDFS we cannot editthe files which are already stored in HDFS, but we can append data by reopening thefiles.



Step 1: The client creates the file by calling create() on DistributedFileSystem(DFS).

Step 2: DFS makes an RPC call to the name node to create a new file in the file system's namespace, with no blocks associated with it. The name node performs various checks to make sure the file doesn't already exist and that the client has the right permissions to create the file. If these checks pass, the name node prepares a record of the new file; otherwise, the file can't be created and therefore the client is thrown an error i.e. IOException. The DFS returns an FSDataOutputStream for the client to start out writing data to.

Step 3: Because the client writes data, the DFSOutputStream splits it into packets, which it writes to an indoor queue called the info queue. The data queue is consumed by the DataStreamer, which is liable for asking the name node to allocate new blocksby picking an inventory of suitable data nodes to store the replicas. The list of data nodes forms a pipeline, and here we'll assume the replication level is three, so there are three nodes in the pipeline. The DataStreamer streams the packets to the primarydata node within the pipeline, which stores each packet and forwards it to the seconddata node within the pipeline.

Step 4: Similarly, the second data node stores the packet and forwards it to the third (and last) data node in the pipeline.

Step 5: The DFSOutputStream sustains an internal queue of packets that are waitingto be acknowledged by data nodes, called an "ack queue".

Step 6: This action sends up all the remaining packets to the data node pipeline and waits for acknowledgments before connecting to the name node to signal whether thefile is complete or not.

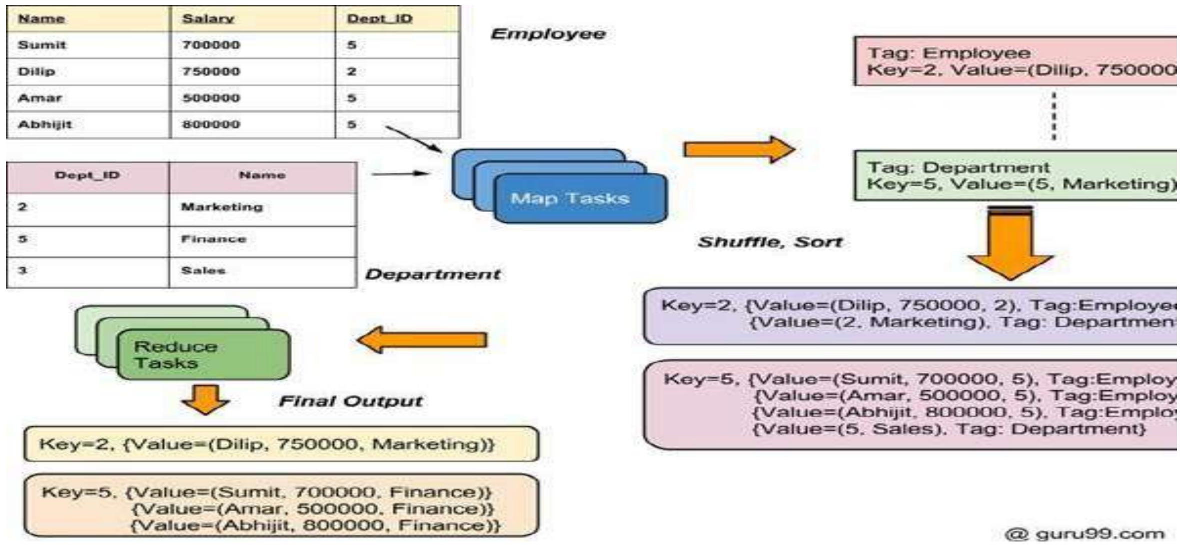
Types of Join:

Depending upon the place where the actual join is performed, joins in Hadoop are classified into-

1. Map-side join – When the join is performed by the mapper, it is called as map-sidejoin. In this type, the join is performed before data is actually consumed by the map function. It is mandatory that the input to each map is in the form of a partition and is in sorted order. Also, there must be an equal number of partitions and it must be sortedby the join key.

2. Reduce-side join – When the join is performed by the reducer, it is called as reduce-side join. There is no necessity in this join to have a dataset in a structured form (or partitioned).

Here, map side processing emits join key and corresponding tuples of both the tables.As an effect of this processing, all the tuples with same join key fall into the same reducer which then joins the records with same join key.An overall process flow of joins in Hadoop is depicted in below diagram.



Name	Salary	Dept_ID
Sumit	700000	5
Dilip	750000	2
Amar	500000	5
Abhijit	800000	5

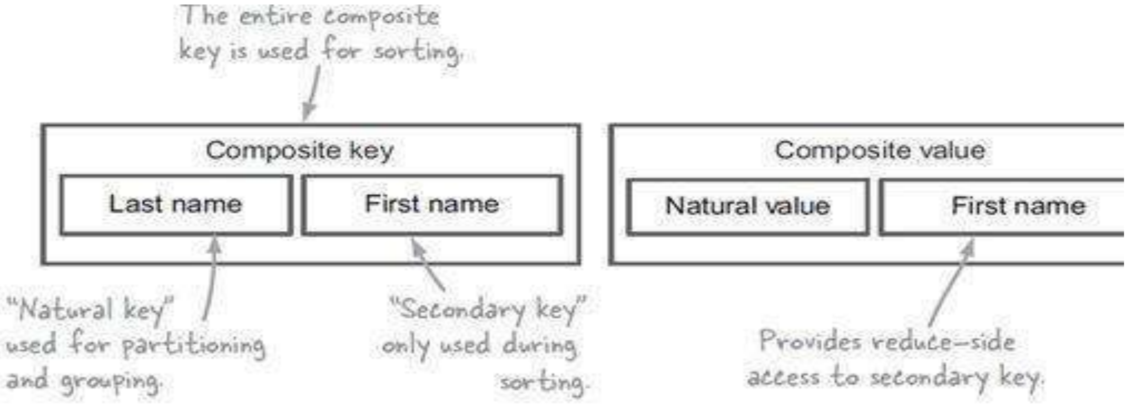
Dept_ID	Name
2	Marketing
5	Finance
3	Sales

Employee
Department
Map Tasks
Tag: Employee
Key=2, Value=(Dilip, 750000
Tag: Department
Key=5, Value=(5, Marketing)
Shuffle, Sort
Key=2, {Value=(Dilip, 750000, 2), Tag:Employe
{Value=(2, Marketing), Tag: Departmen
Key=5, {Value=(Sumit, 700000, 5), Tag:Employ
{Value=(Amar, 500000, 5), Tag:Employ
{Value=(Abhijit, 800000, 5), Tag:Emplo
{Value=(5, Sales), Tag: Department}
Reduce Tasks
Final Output
Key=2, {Value=(Dilip, 750000, Marketing)}
Key=5, {Value=(Sumit, 700000, Finance)}
{Value=(Amar, 500000, Finance)}
{Value=(Abhijit, 800000, Finance)}
@ guru99.com

Secondary Sorting:

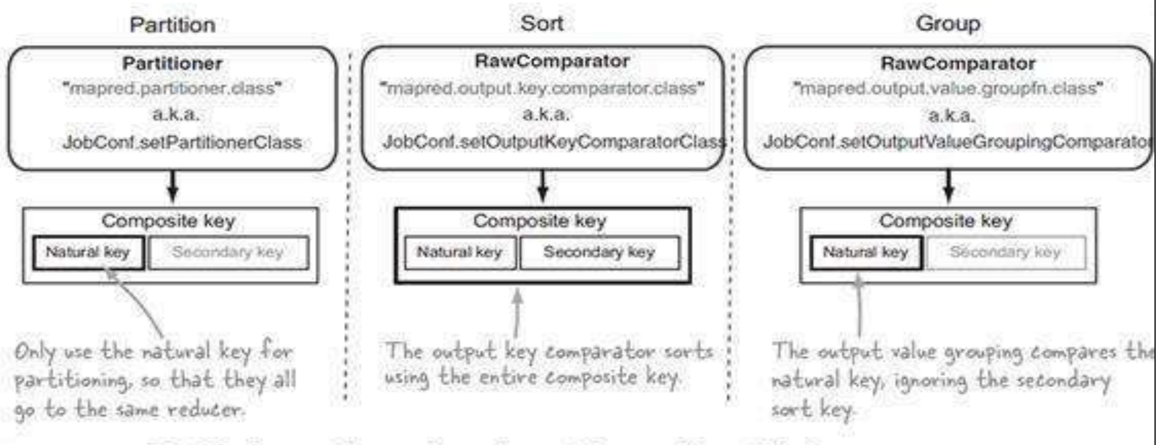
Secondary sort is a technique that allows the MapReduce programmer to control the order that the values show up within a reduce function call.

Lets also assume that our secondary sorting is on a composite key made out of Last Name and First Name.


The entire composite key is used for sorting.
Composite key
Last name
First name
Composite value
Natural value
First name
"Natural key" used for partitioning and grouping.
"Secondary key" only used during sorting.
Provides reduce-side access to secondary key.

The user composite key and value

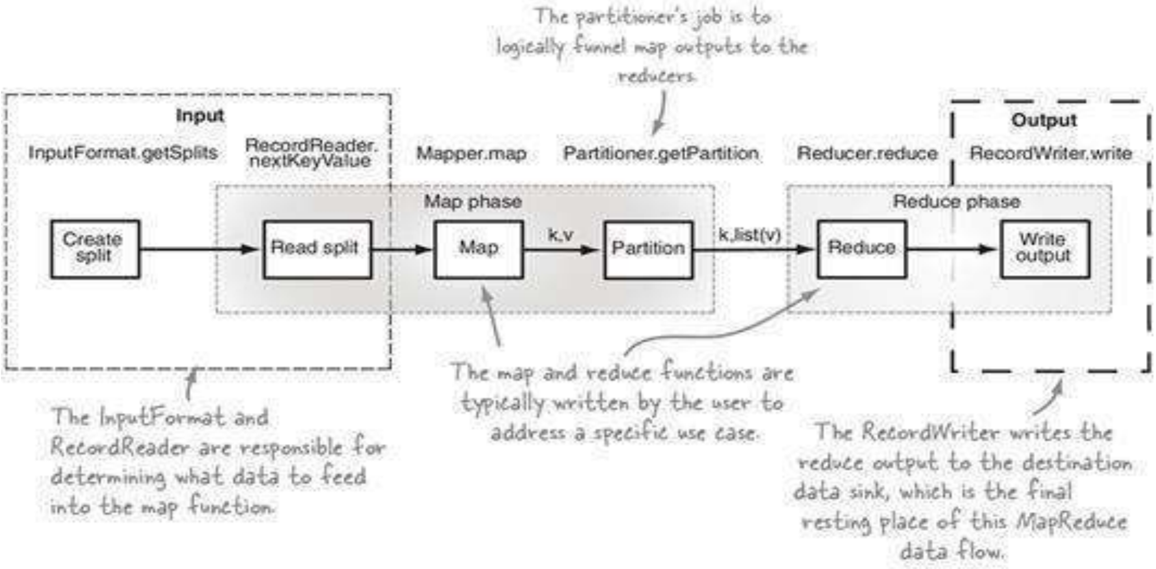
Now lets look at the steps involved in secondary sorting


Partition
Partitioner
"mapred.partitioner.class"
a.k.a.
JobConf.setPartitionerClass
Composite key
Natural key
Secondary key
Only use the natural key for partitioning, so that they all go to the same reducer.
Sort
RawComparator
"mapred.output.key.comparator.class"
a.k.a.
JobConf.setOutputKeyComparatorClass
Composite key
Natural key
Secondary key
The output key comparator sorts using the entire composite key.
Group
RawComparator
"mapred.output.value.groupfn.class"
a.k.a.
JobConf.setOutputValueGroupingComparator
Composite key
Natural key
Secondary key
The output value grouping compares the natural key, ignoring the secondary sort key.

Partitioning, sorting, and grouping settings and key utilization

The partitioner and the group comparator use only **natural key**, the partitioner uses itto channel all records with the same natural key to a single reducer. This partitioning happens in the Map Phase, data from various Map tasks are received by reducers where they are **grouped** and then sent to the reduce method. This grouping is wherethe group comparator comes into picture, if we would **not** have specified a custom group comparator then Hadoop would have used the default implementation which would have considered the entire composite key, which would have lead to incorrect results.

Finally just reviewing the steps involved in a MR Job and relating it to secondary sorting should help us clear out the lingering doubts.



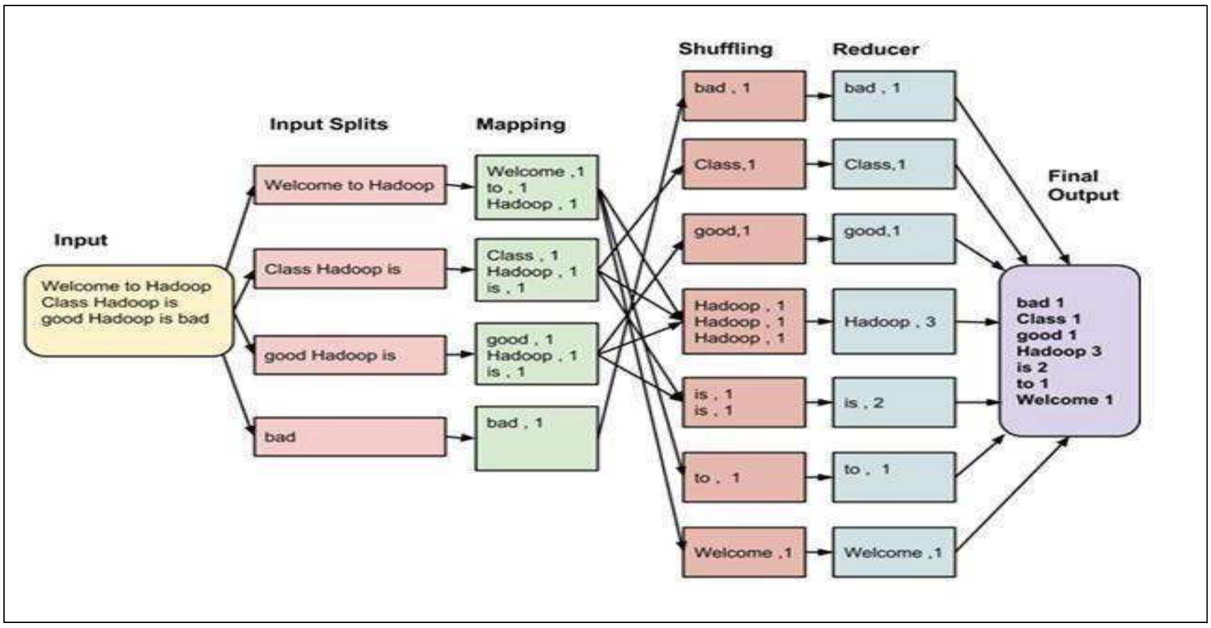
The partitioner's job is to logically funnel map outputs to the reducers.

The InputFormat and RecordReader are responsible for determining what data to feed into the map function.

The map and reduce functions are typically written by the user to address a specific use case.

The RecordWriter writes the reduce output to the destination data sink, which is the final resting place of this MapReduce data flow.

Pipelining Map Reduce jobs:


Input
Welcome to Hadoop Class Hadoop is good Hadoop is bad
Input Splits
Welcome to Hadoop
Class Hadoop is
good Hadoop is
bad
Mapping
Welcome , 1 to , 1 Hadoop , 1
Class , 1 Hadoop , 1 is , 1
good , 1 Hadoop , 1 is , 1
bad , 1
Shuffling
bad , 1
Class,1
good,1
Hadoop , 1 Hadoop , 1 Hadoop , 1
is , 1 is , 1
to , 1
Welcome ,1
Reducer
bad , 1
Class,1
good,1
Hadoop , 3
is , 2
to , 1
Welcome ,1
Final Output
bad 1 Class 1 good 1 Hadoop 3 is 2 to 1 Welcome 1

Introduction to Apache spark

What is Spark?

Apache Spark is an open-source cluster computing framework. Its primary purpose isto handle the real-time generated data.

Spark was built on the top of the Hadoop MapReduce. It was optimized to run in memory whereas alternative approaches like Hadoop's MapReduce writes data to andfrom computer hard drives. So, Spark process the data much quicker than other alternatives.

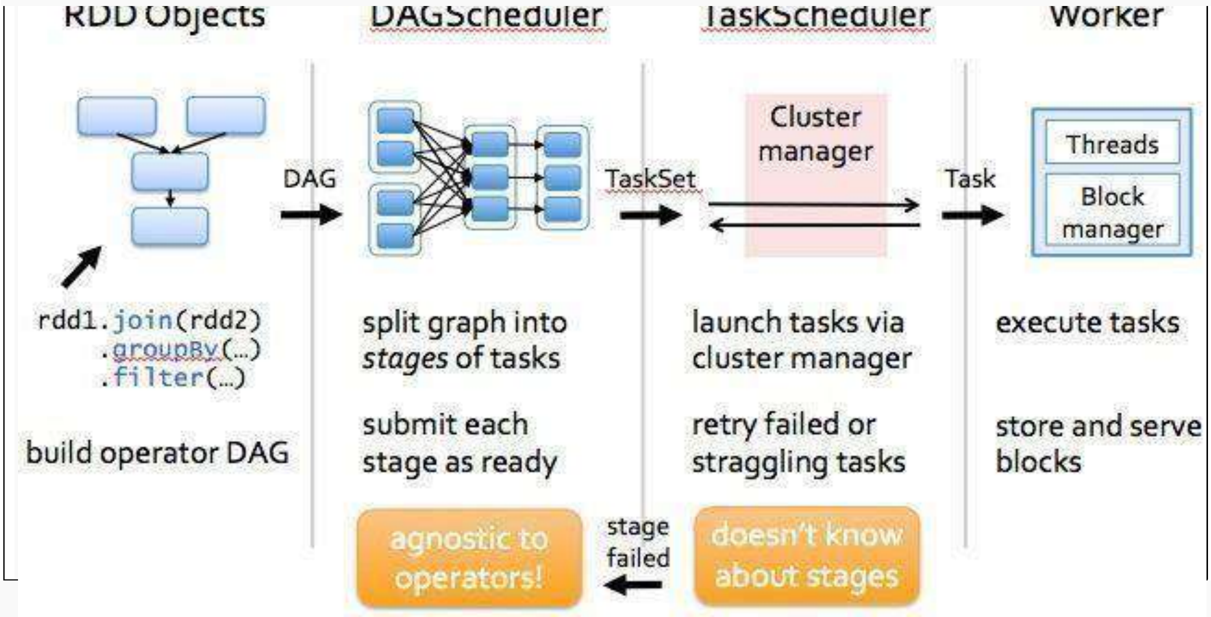
Features of Apache Spark

- **Fast** - It provides high performance for both batch and streaming data, using astate-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.
- **Easy to Use** - It facilitates to write the application in Java, Scala, Python, R, and SQL. It also provides more than 80 high-level operators.
- **Generality** - It provides a collection of libraries including SQL and DataFrames,MLlib for machine learning, GraphX, and Spark Streaming.
- **Lightweight** - It is a light unified analytics engine which is used for large scaledata processing.
- **Runs Everywhere** - It can easily run on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud.

5.3 How spark works

Spark has a small code base and the system is divided in various layers. Each layerhas some responsibilities. The layers are independent of each other.

The first layer is the interpreter, Spark uses a Scala interpreter, with somemodifications. As you enter your code in spark console (creating RDD's and applying operators), Spark creates a operator graph. When the user runs an action (like collect) , the Graph is submitted to a DAG Scheduler. The DAG scheduler divides operator graph into (map and reduce) stages. A stage is comprised of tasks based on partitions of the input data. The DAG scheduler pipelines operators together to optimize the graph. For e.g. Many map operators can be scheduled in a single stage. This optimization is key to Sparks performance. The final result of a DAG scheduler is a set of stages. The stages are passed on to the Task Scheduler. The task scheduler launches tasks via cluster manager. (Spark Standalone/Yarn/Mesos). The task scheduler doesn't know about dependencies among stages.



What is RDD?

The RDD (Resilient Distributed Dataset) is the Spark's core abstraction. It is acollection of elements, partitioned across the nodes of the cluster so that we can execute various parallel operations on it.

There are two ways to create RDDs:

- Parallelizing an existing data in the driver program
- Referencing a dataset in an external storage system, such as a shared filesystem, HDFS, HBase, or any data source offering a Hadoop Input Format.

RDD Operations

The RDD provides the two types of operations:

- Transformation
- Action

Transformation

In Spark, the role of transformation is to create a new dataset from an existing one. The transformations are considered lazy as they only computed when an action requires a result to be returned to the driver program.

Let's see some of the frequently used RDD Transformations.

map(func)	It returns a new distributed dataset formed by passing eachelement of the source through a function func.
filter(func)	It returns a new dataset formed by selecting those elements ofthe source on which func returns true.
flatMap(func)	Here, each input item can be mapped to zero or more output items, so func should return a sequence rather than a single item.
mapPartitions(func)	It is similar to map, but runs separately on each partition (block)of the RDD, so func must be of type Iterator<T> => Iterator<U> when running on an RDD of type T.

Map Partitions With Index (func)	It is similar to map Partitions that provides func with an integer value representing the index of the partition, so func must be of

	type (Int, Iterator<T>) => Iterator<U> when running on an RDD of type T.
sample(withReplacement , fraction, seed)	It samples the fraction fraction of the data, with or without replacement, using a given random number generator seed.
union(otherDataset)	It returns a new dataset that contains the union of the elementsin the source dataset and the argument.
intersection(otherDataset)	It returns a new RDD that contains the intersection of elementsin the source dataset and the argument.
distinct([numPartitions]))	It returns a new dataset that contains the distinct elements ofthe source dataset.
groupByKey([numPartitions])	It returns a dataset of (K, Iterable) pairs when called on adataset of (K, V) pairs.
reduceByKey(func ,[numPartitions])	When called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated usingthe given reduce function func, which must be of type (V,V) => V.
aggregateByKey(zeroValue)(seqOp, combOp, [numPartitions])	When called on a dataset of (K, V) pairs, returns a dataset of (K, U) pairs where the values for each key are aggregated usingthe given combine functions and a neutral "zero" value.
sortByKey([ascending], [numPartitions])	It returns a dataset of key-value pairs sorted by keys in ascending or descending order, as specified in the boolean ascending argument.

join(otherDataset, [numPartitions])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.
cogroup(otherDataset, [numPartitions])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (Iterable, Iterable)) tuples. This operation is also called groupWith.
cartesian(otherDataset)	When called on datasets of types T and U, returns a dataset of (T, U) pairs (all pairs of elements).
pipe(command, [envVars])	Pipe each partition of the RDD through a shell command, e.g. a Perl or bash script.
coalesce(numPartitions)	It decreases the number of partitions in the RDD to numPartitions.
repartition(numPartitions)	It reshuffles the data in the RDD randomly to create either more or fewer partitions and balance it across them.
repartitionAndSortWithinPartitions(partitioner)	It repartition the RDD according to the given partitioner and, within each resulting partition, sort records by their keys.

Action

In Spark, the role of action is to return a value to the driver program after running acomputation on the dataset.

Let's see some of the frequently used RDD Actions.

Action	Description

reduce(func)	It aggregate the elements of the dataset using a function func (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.
collect()	It returns all the elements of the dataset as an array at thedriver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count()	It returns the number of elements in the dataset.
first()	It returns the first element of the dataset (similar totake(1)).
take(n)	It returns an array with the first n elements of the dataset.
takeSample(withReplacement, num, [seed])	It returns an array with a random sample of num elementsof the dataset, with or without replacement, optionally pre-specifying a random number generator seed.
takeOrdered(n, [ordering])	It returns the first n elements of the RDD using either their natural order or a custom comparator.
saveAsTextFile(path)	It is used to write the elements of the dataset as a text file(or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark calls toString on each element to convert itto a line of text in the file.

saveAsSequenceFile(path)(Java and Scala)	It is used to write the elements of the dataset as a Hadoop SequenceFile in a given path in the local filesystem, HDFSor any other Hadoop-supported file system.
saveAsObjectFile(path)(Java and Scala)	It is used to write the elements of the dataset in a simple format using Java serialization, which can then be loaded usingSparkContext.objectFile().
countByKey()	It is only available on RDDs of type (K, V). Thus, it returnsa hashmap of (K, Int) pairs with the count of each key.
foreach(func)	It runs a function func on each element of the dataset for side effects such as updating an Accumulator or interacting with external storage systems.

*****