MODULE 1

1.1 CLASSIFICATION OF DIGITAL DATA

As depicted in Figure 1.1, digital data can be broadly classified into structured, semi-structured, and unstructured data.

1. **Unstructured data:** This is the data which does not conform to a data model or is not in a form which can be used easily by a computer program. About 80–90% data of an organization is in this format; for example, memos, chat rooms, PowerPoint presentations, images, videos, letters, researches, white papers, body of an email, etc.
2. **Semi-structured data:** This is the data which does not conform to a data model but has some structure. However, it is not in a form which can be used easily by a computer program; for example, emails, XML, markup languages like HTML, etc. Metadata for this data is available but is not sufficient.
3. **Structured data:** This is the data which is in an organized form (e.g., in rows and columns) and can be easily used by a computer program. Relationships exist between entities of data, such as classes and their objects. Data stored in databases is an example of structured data.

Ever since the 1980s most of the enterprise data has been stored in relational databases complete with rows/records/tuples, columns/attributes/fields, primary keys, foreign keys, etc. Over a period of time Relational Database Management System (RDBMS) matured and the RDBMS, as they are available today, have become more robust, cost-effective, and efficient. We have grown comfortable working with RDBMS – the storage, retrieval, and management of data has been immensely simplified. The data held in RDBMS is typically structured data. However, with the Internet connecting the world, data that existed beyond one's enterprise started to become an integral part of daily transactions. This data grew by leaps and bounds so much so that it became difficult for the enterprises to ignore it. All of this data was not structured. A lot of it was unstructured. In fact, Gartner estimates that almost 80% of data generated in any enterprise today is unstructured data. Roughly around 10% of data is in the structured and semi-structured category. Refer Figure 1.2.

1.1.1 Structured Data

Let us begin with a very basic question – When do we say that the data is structured? The simple answer is when data conforms to a pre-defined schema/structure we say it is structured data.

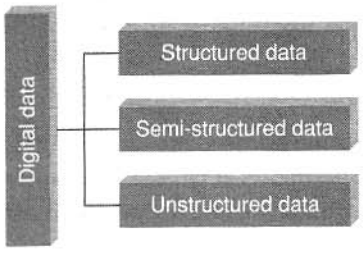


Figure 1.1 Classification of digital data.

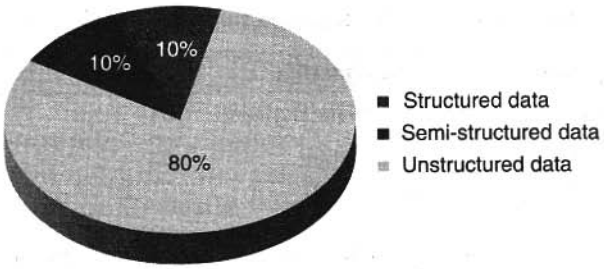


Figure 1.2 Approximate percentage distribution of digital data.

Think structured data, and think data model – a model of the types of business data that we intend to store, process, and access. Let us discuss this in the context of an RDBMS. Most of the structured data is held in RDBMS. An RDBMS conforms to the relational data model wherein the data is stored in rows/columns. Refer Table 1.1.

The number of rows/records/tuples in a relation is called the *cardinality of a relation* and the number of columns is referred to as the *degree of a relation.*

The first step is the design of a relation/table, the fields/columns to store the data, the type of data that will be stored [number (integer or real), alphabets, date, Boolean, etc.]. Next we think of the constraints that we would like our data to conform to (constraints such as UNIQUE values in the column, NOT NULL values in the column, a business constraint such as the value held in the column should not drop below 50, the set of permissible values in the column such as the column should accept only "CS", "IS", "MS", etc., as input).

To explain further, let us design a table/relation structure to store the details of the employees of an enterprise. Table 1.2 shows the structure/schema of an "Employee" table in a RDBMS such as Oracle.

Table 1.2 is an example of a good structured table (complete with table name, meaningful column names with data types, data length, and the relevant constraints) with absolute adherence to relational data model.

Table 1.1 A relation/table with rows and columns

	Column 1	Column 2	Column 3	Column 4
Row 1				

Table 1.2 Schema of an "Employee" table in a RDBMS such as Oracle

Column Name	Data Type	Constraints
EmpNo	Varchar(10)	PRIMARY KEY
EmpName	Varchar(50)	
Designation	Varchar(25)	NOT NULL
DeptNo	Varchar(5)	
ContactNo	Varchar(10)	NOT NULL

Table 1.3 Sample records in the "Employee" table

EmpNo	EmpName	Designation	DeptNo	ContactNo
E101	Allen	Software Engineer	D1	0999999999
E102	Simon	Consultant	D1	0777777777

It goes without saying that each record in the table will have exactly the same structure. Let us take a look at a few records in Table 1.3.

The tables in an RDBMS can also be related. For example, the above "Employee" table is related to the "Department" table on the basis of the common column, "DeptNo". It is not mandatory for the two tables that are related to have exactly the same name for the common column. On the contrary, the two tables are related on the basis of values held within the column, "DeptNo". Given in Figure 1.3 is a depiction of referential integrity constraint (primary – foreign key) with the "Department" table being the referenced table and "Employee" table being the referencing table.

1.1.1.1 Sources of Structured Data

If your data is highly structured, one can look at leveraging any of the available RDBMS [Oracle Corp. – Oracle, IBM – DB2, Microsoft – Microsoft SQL Server, EMC – Greenplum, Teradata – Teradata, MySQL (open source), PostgreSQL (advanced open source), etc.] to house it. Refer Figure 1.4. These databases are typically used to hold transaction/operational data generated and collected by day-to-day business activities. In other words, the data of the On-Line Transaction Processing (OLTP) systems are generally quite structured.

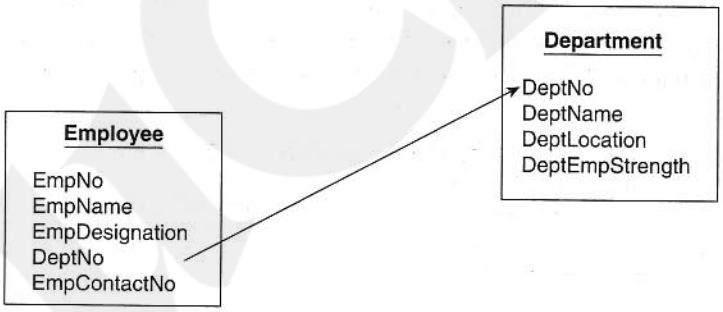


Figure 1.3 Relationship between "Employee" and "Department" tables.

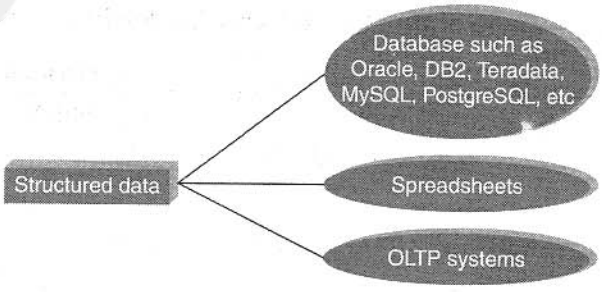


Figure 1.4 Sources of structured data.

1.1.1.2 Ease of Working with Structured Data

Structured data provides the ease of working with it. Refer Figure 1.5. The ease is with respect to the following:

1. **Insert/update/delete:** The Data Manipulation Language (DML) operations provide the required ease with data input, storage, access, process, analysis, etc.
2. **Security:** How does one ensure the security of information? There are available staunch encryption and tokenization solutions to warrant the security of information throughout its lifecycle. Organizations are able to retain control and maintain compliance adherence by ensuring that only authorized individuals are able to decrypt and view sensitive information.
3. **Indexing:** An index is a data structure that speeds up the data retrieval operations (primarily the SELECT DML statement) at the cost of additional writes and storage space, but the benefits that ensue in search operation are worth the additional writes and storage space.
4. **Scalability:** The storage and processing capabilities of the traditional RDBMS can be easily scaled up by increasing the horsepower of the database server (increasing the primary and secondary or peripheral storage capacity, processing capacity of the processor, etc.).
5. **Transaction processing:** RDBMS has support for Atomicity, Consistency, Isolation, and Durability (ACID) properties of transaction. Given next is a quick explanation of the ACID properties:
   - ***Atomicity:*** A transaction is atomic, means that either it happens in its entirety or none of it at all.
   - ***Consistency:*** The database moves from one consistent state to another consistent state. In other words, if the same piece of information is stored at two or more places, they are in complete agreement.
   - ***Isolation:*** The resource allocation to the transaction happens such that the transaction gets the impression that it is the only transaction happening in isolation.
   - ***Durability:*** All changes made to the database during a transaction are permanent and that accounts for the durability of the transaction.

1.1.2 Semi-Structured Data

Semi-structured data is also referred to as self-describing structure. Refer Figure 1.6. It has the following features:

1. It does not conform to the data models that one typically associates with relational databases or any other form of data tables.
2. It uses tags to segregate semantic elements.

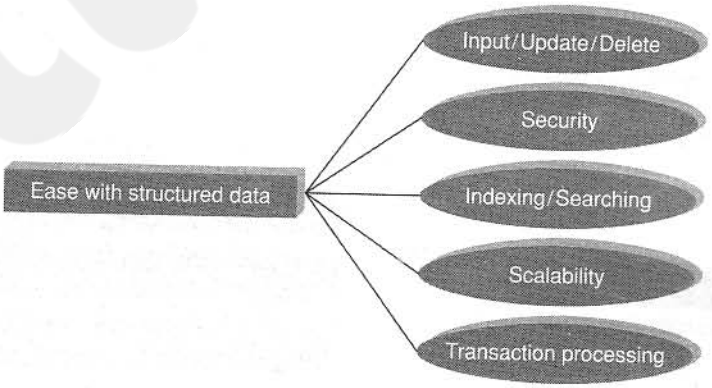


Figure 1.5 Ease of working with structured data.

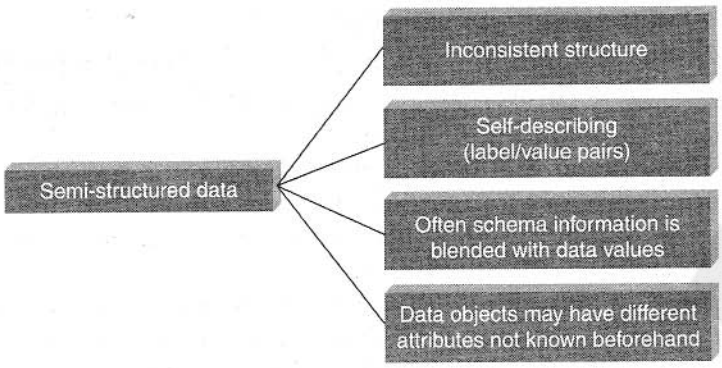


Figure 1.6 Characteristics of semi-structured data.

3. Tags are also used to enforce hierarchies of records and fields within data.
4. There is no separation between the data and the schema. The amount of structure used is dictated by the purpose at hand.
5. In semi-structured data, entities belonging to the same class and also grouped together need not necessarily have the same set of attributes. And if at all, they have the same set of attributes, the order of attributes may not be similar and for all practical purposes it is not important as well.

1.1.2.1 Sources of Semi-Structured Data

Amongst the sources for semi-structured data, the front runners are "XML" and "JSON" as depicted in Figure 1.7.

1. **XML:** eXtensible Markup Language (XML) is hugely popularized by web services developed utilizing the Simple Object Access Protocol (SOAP) principles.
2. **JSON:** Java Script Object Notation (JSON) is used to transmit data between a server and a web application. JSON is popularized by web services developed utilizing the Representational State Transfer (REST) – an architecture style for creating scalable web services. MongoDB (open-source, distributed, NoSQL, documented-oriented database) and Couchbase (originally known as Membase, open-source, distributed, NoSQL, document-oriented database) store data natively in JSON format.

An example of HTML is as follows:

```
<HTML>
<HEAD>
<TITLE>Place your title here</TITLE>
</HEAD>
<BODY BGCOLOR="FFFFFF">
```

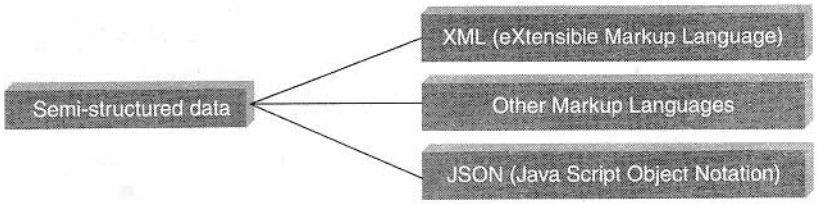


Figure 1.7 Sources of semi-structured data.

```
<CENTER><IMG SRC="clouds.jpg" ALIGN="BOTTOM"></CENTER>
<HR>
<a href="http://bigdatauniversity.com">Link Name</a>
<H1>this is a Header</H1>
<H2>this is a sub Header</H2>
Send me mail at <a href="mailto:support@yourcompany.com">
support@yourcompany.com</a>.
<P>a new paragraph!
<P><B>a new paragraph!</B>
<BR><B><I>this is a new sentence without a paragraph break, in bold italics.</I></B>
<HR>
</BODY>
</HTML>
```

Sample JSON document

```
{
_id:9,
BookTitle: "Fundamentals of Business Analytics",
AuthorName: "Seema Acharya",
Publisher: "Wiley India",
YearofPublication: "2011"
}
```

1.1.3 Unstructured Data

Unstructured data does not conform to any pre-defined data model. In fact, to explain things a little more, let us take a closer look at the various kinds of text available and the possible structure associated with it. As can be seen from the examples quoted in Table 1.4, the structure is quite unpredictable. In Figure 1.8 we look at the other sources of unstructured data.

1.1.3.1 *Issues with "Unstructured" Data*

Although unstructured data is known NOT to conform to a pre-defined data model or be organized in a pre-defined manner, there are incidents wherein the structure of the data (placed in the unstructured category) can still be implied. As mentioned in Figure 1.9, there could be few other reasons behind placing data in the unstructured category despite it having some structure or being highly structured.

There are situations where people argue that a text file should be in the category of semi-structured data and not unstructured data. Let us look at where they are coming from. Well, the text file does have a name,

Table 1.4 Few examples of disparate unstructured data

Twitter message	Feeling miffed⊗. Victim of twishing.
Facebook post	LOL. C ya. BFN
Log files	127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326 "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I; Nav)"
Email	Hey Joan, possible to send across the first cut on the Hadoop chapter by Friday EOD or maybe we can meet up over a cup of coffee. Best regards, Tom

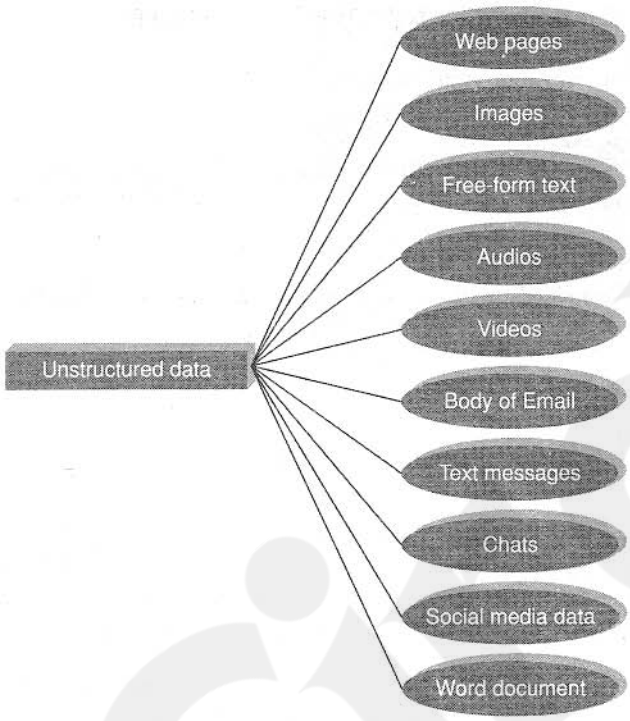


Figure 1.8 Sources of unstructured data.

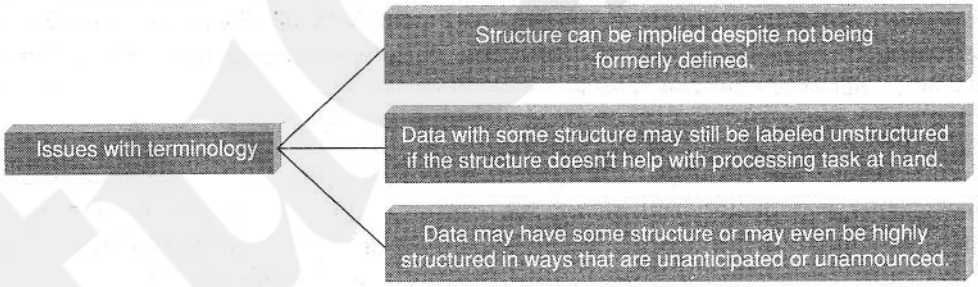


Figure 1.9 Issues with terminology of unstructured data.

one can easily look at the properties to get information such as the owner of the file, the date on which the file was created, the size of the file, etc. Okay, we do have little metadata. But when it comes to analysis, we are more concerned with the content of the text file rather than the name or any of the other properties. In fact, the other properties may not in any way contribute to the processing/analysis task at hand. Therefore, it is fair to place it in the unstructured data category.

1.1.3.2 How to Deal with Unstructured Data?

Today, unstructured data constitutes approximately 80% of the data that is being generated in any enterprise. The balance is clearly shifting in favor of unstructured data as shown in Figure 1.10. It is such a big percentage that it cannot be ignored. Figure 1.11 states a few ways of dealing with unstructured data.

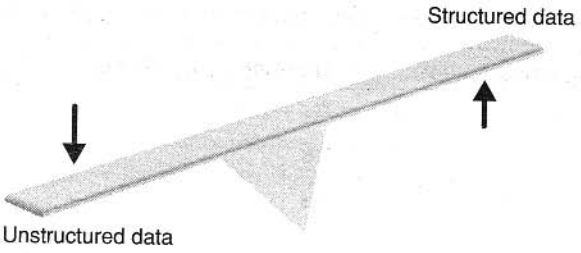


Figure 1.10 Unstructured data clearly constitutes a major percentage of enterprise data.

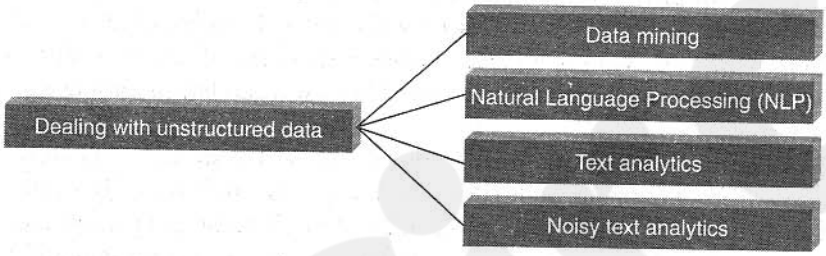


Figure 1.11 Dealing with unstructured data.

The following techniques are used to find patterns in or interpret unstructured data:

1. **Data mining:** First, we deal with large data sets. Second, we use methods at the intersection of artificial intelligence, machine learning, statistics, and database systems to unearth consistent patterns in large data sets and/or systematic relationships between variables. It is the analysis step of the "knowledge discovery in databases" process.

   Few popular data mining algorithms are as follows:

   - ***Association rule mining:*** It is also called "market basket analysis" or "affinity analysis". It is used to determine "What goes with what?" It is about when you buy a product, what is the other product that you are likely to purchase with it. For example, if you pick up bread from the grocery, are you likely to pick eggs or cheese to go with it.
   - ***Regression analysis:*** It helps to predict the relationship between two variables. The variable whose value needs to be predicted is called the dependent variable and the variables which are used to predict the value are referred to as the independent variables.

PICTURE THIS...

You are interested in purchasing real estate. You have been looking at a few good sites. You have come to the conclusion that cost of the real estate depends on the location (outskirts or prime locale), the amenities provided by the builder (joggers track, senior citizen zone, gymnasium, swimming pools, etc.), the built up area, etc. The cost of the real estate is the dependent variable and the location, amenities, built-up area are called the independent variables.

Table 1.5 Sample records depicting learners' preferences for modes of learning

	Learning using Audios	Learning using Videos	Textual Learners
User 1	Yes	Yes	No
User 2	Yes	Yes	Yes
User 3	Yes	Yes	No
User 4	Yes	?	?

- ***Collaborative filtering:*** It is about predicting a user's preference or preferences based on the preferences of a group of users. For example, take a look at Table 1.5.
  We are looking at predicting whether User 4 will prefer to learn using videos or is a textual learner depending on one or a couple of his or her known preferences. We analyze the preferences of similar user profiles and on the basis of it, predict that User 4 will also like to learn using videos and is not a textual learner.

2. **Text analytics or text mining:** Compared to the structured data stored in relational databases, text is largely unstructured, amorphous, and difficult to deal with algorithmically. Text mining is the process of gleaning high quality and meaningful information (through devising of patterns and trends by means of statistical pattern learning) from text. It includes tasks such as text categorization, text clustering, sentiment analysis, concept/entity extraction, etc.
3. **Natural language processing (NLP):** It is related to the area of human computer interaction. It is about enabling computers to understand human or natural language input.
4. **Noisy text analytics:** It is the process of extracting structured or semi-structured information from noisy unstructured data such as chats, blogs, wikis, emails, message-boards, text messages, etc. The noisy unstructured data usually comprises one or more of the following: Spelling mistakes, abbreviations, acronyms, non-standard words, missing punctuation, missing letter case, filler words such as "uh", "um", etc.
5. **Manual tagging with metadata:** This is about tagging manually with adequate metadata to provide the requisite semantics to understand unstructured data.
6. **Part-of-speech tagging:** It is also called POS or POST or grammatical tagging. It is the process of reading text and tagging each word in the sentence as belonging to a particular part of speech such as "noun", "verb", "adjective", etc.
7. **Unstructured Information Management Architecture (UIMA):** It is an open source platform from IBM. It is used for real-time content analytics. It is about processing text and other unstructured data to find latent meaning and relevant relationship buried therein. Read up more on UIMA at the link: http://www.ibm.com/developerworks/data/downloads/uima/

2.1 CHARACTERISTICS OF DATA

Let us start with the characteristics of data. As depicted in Figure 2.1, data has three key characteristics:

1. **Composition:** The composition of data deals with the structure of data, that is, the sources of data, the granularity, the types, and the nature of data as to whether it is static or real-time streaming.
2. **Condition:** The condition of data deals with the state of data, that is, "Can one use this data as is for analysis?" or "Does it require cleansing for further enhancement and enrichment?"
3. **Context:** The context of data deals with "Where has this data been generated?" "Why was this data generated?" "How sensitive is this data?" "What are the events associated with this data?" and so on.

Small data (data as it existed prior to the big data revolution) is about certainty. It is about fairly known data sources; it is about no major changes to the composition or context of data.

Most often we have answers to queries like why this data was generated, where and when it was generated, exactly how we would like to use it, what questions will this data be able to answer, and so on. Big data is

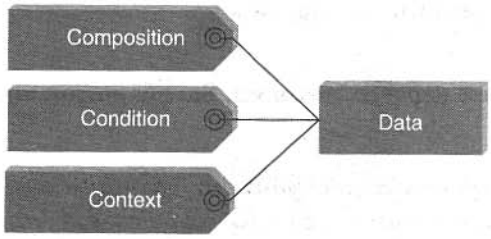


Figure 2.1 Characteristics of data.

about complexity... complexity in terms of multiple and unknown datasets, in terms of exploding volume, in terms of the speed at which the data is being generated and the speed at which it needs to be processed, and in terms of the variety of data (internal or external, behavioral or social) that is being generated.

2.2 EVOLUTION OF BIG DATA

1970s and before was the era of mainframes. The data was essentially primitive and structured. Relational databases evolved in 1980s and 1990s. The era was of data intensive applications. The World Wide Web (WWW) and the Internet of Things (IoT) have led to an onslaught of structured, unstructured, and multimedia data. Refer Table 2.1.

Table 2.1 The evolution of big data

	Data Generation and Storage	Data Utilization	Data Driven
Complex and Unstructured			Structured data, unstructured data, multimedia data
Complex and Relational		Relational databases: Data-intensive applications	
Primitive and Structured	Mainframes: Basic data storage		
	1970s and before	**Relational (1980s and 1990s)**	**2000s and beyond**

2.3 DEFINITION OF BIG DATA

If we were to ask you the simple question: "Define Big Data", what would your answer be? Well, we will give you a few responses that we have heard over time:

1. Anything beyond the human and technical infrastructure needed to support storage, processing, and analysis.
2. Today's BIG may be tomorrow's NORMAL.

3. Terabytes or petabytes or zettabytes of data.
4. I think it is about 3 Vs.

Refer Figure 2.2. Well, all of these responses are correct. But it is not just one of these; in fact, big data is all of the above and more.

> *Big data is high-volume, high-velocity, and high-variety information assets that demand cost effective, innovative forms of information processing for enhanced insight and decision making.*
>
> *Source:* Gartner IT Glossary

The 3Vs concept was proposed by the Gartner analyst Doug Laney in a 2001 MetaGroup research publication, titled, *3D Data Management: Controlling Data Volume, Variety and Velocity.*

Source: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

For the sake of easy comprehension, we will look at the definition in three parts. Refer Figure 2.3.

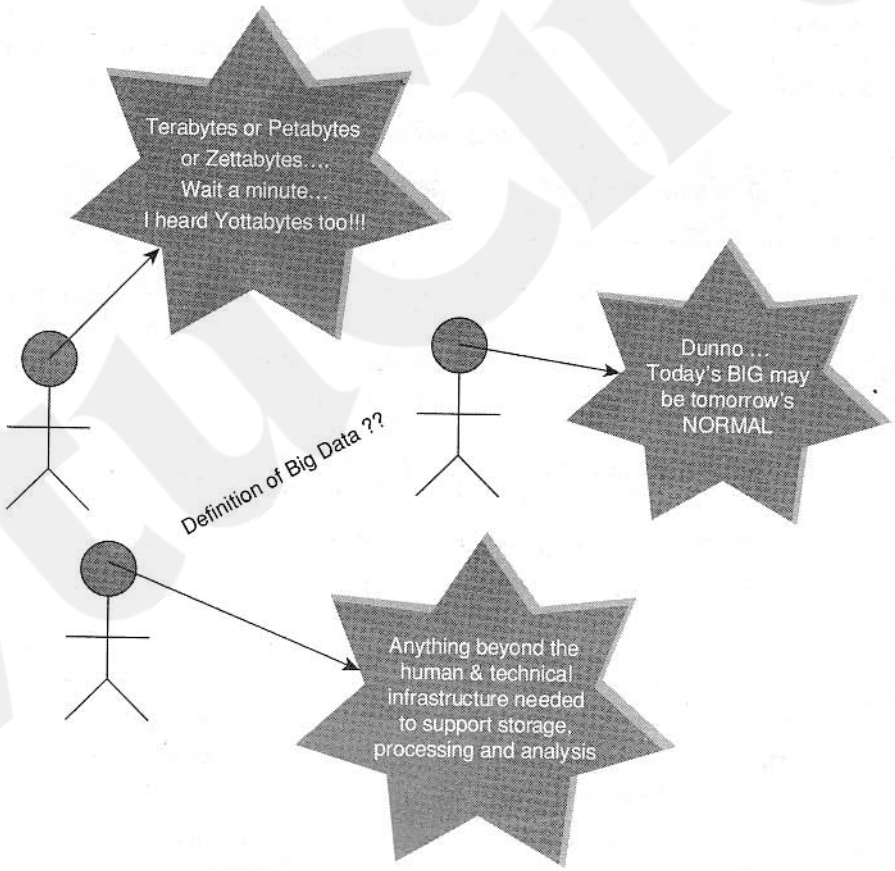


Figure 2.2 Definition of big data.

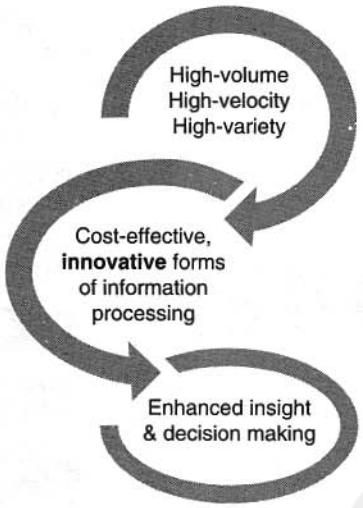


Figure 2.3 Definition of big data – Gartner.

Part I of the definition "big data is high-volume, high-velocity, and high-variety information assets" talks about voluminous data (humongous data) that may have great variety (a good mix of structured, semi-structured, and unstructured data) and will require a good speed/pace for storage, preparation, processing, and analysis.

Part II of the definition "cost effective, innovative forms of information processing" talks about embracing new techniques and technologies to capture (ingest), store, process, persist, integrate, and visualize the high-volume, high-velocity, and high-variety data.

Part III of the definition "enhanced insight and decision making" talks about deriving deeper, richer, and meaningful insights and then using these insights to make faster and better decisions to gain business value and thus a competitive edge.

Data → Information → Actionable intelligence → Better decisions → Enhanced business value

2.4 CHALLENGES WITH BIG DATA

Refer Figure 2.4. Following are a few challenges with big data:

1. Data today is growing at an exponential rate. Most of the data that we have today has been generated in the last 2–3 years. This high tide of data will continue to rise incessantly. The key questions here are: "Will all this data be useful for analysis?", "Do we work with all this data or a subset of it?", "How will we separate the knowledge from the noise?", etc.
2. Cloud computing and virtualization are here to stay. Cloud computing is the answer to managing infrastructure for big data as far as cost-efficiency, elasticity, and easy upgrading/downgrading is concerned. This further complicates the decision to host big data solutions outside the enterprise.
3. The other challenge is to decide on the period of retention of big data. Just how long should one retain this data? A tricky question indeed as some data is useful for making long-term decisions, whereas in few cases, the data may quickly become irrelevant and obsolete just a few hours after having being generated.

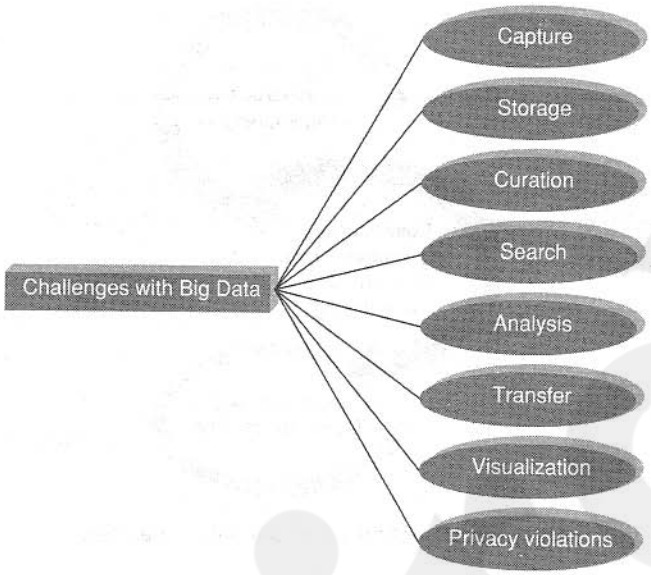


Figure 2.4 Challenges with big data.

4. There is a dearth of skilled professionals who possess a high level of proficiency in data sciences that is vital in implementing big data solutions.
5. Then, of course, there are other challenges with respect to capture, storage, preparation, search, analysis, transfer, security, and visualization of big data. Big data refers to datasets whose size is typically beyond the storage capacity of traditional database software tools. There is no explicit definition of how big the dataset should be for it to be considered "big data." Here we are to deal with data that is just too big, moves way to fast, and does not fit the structures of typical database systems. The data changes are highly dynamic and therefore there is a need to ingest this as quickly as possible.
6. Data visualization is becoming popular as a separate discipline. We are short by quite a number, as far as business visualization experts are concerned.

2.5 WHAT IS BIG DATA?

Big data is data that is big in volume, velocity, and variety. Refer Figure 2.5.

2.5.1 Volume

We have seen it grow from bits to bytes to petabytes and exabytes. Refer Table 2.2 and Figure 2.6.

Bits → Bytes → Kilobytes → Megabytes → Gigabytes → Terabytes → Petabytes → Exabytes → Zettabytes → Yottabytes

2.5.1.1 Where Does This Data get Generated?

There are a multitude of sources for big data. An XLS, a DOC, a PDF, etc. is unstructured data; a video on YouTube, a chat conversation on Internet Messenger, a customer feedback form on an online retail website

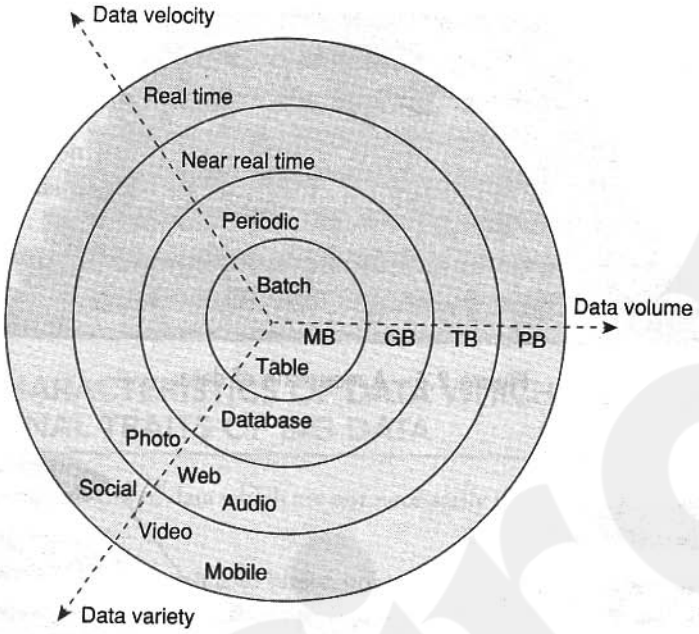


Figure 2.5 Data: Big in volume, variety, and velocity.

Table 2.2 Growth of data

Bits	0 or 1
Bytes	8 bits
Kilobytes	1024 bytes
Megabytes	1024^2 bytes
Gigabytes	1024^3 bytes
Terabytes	1024^4 bytes
Petabytes	1024^5 bytes
Exabytes	1024^6 bytes
Zettabytes	1024^7 bytes
Yottabytes	1024^8 bytes

unstructured data; a CCTV coverage, a weather forecast report is unstructured data too. Refer Figure 2.7 for the sources of big data.

1. **Typical internal data sources:** Data present within an organization's firewall. It is as follows:
   - ***Data storage:*** File systems, SQL (RDBMSs – Oracle, MS SQL Server, DB2, MySQL, PostgreSQL, etc.), NoSQL (MongoDB, Cassandra, etc.), and so on.
   - ***Archives:*** Archives of scanned documents, paper archives, customer correspondence records, patients' health records, students' admission records, students' assessment records, and so on.

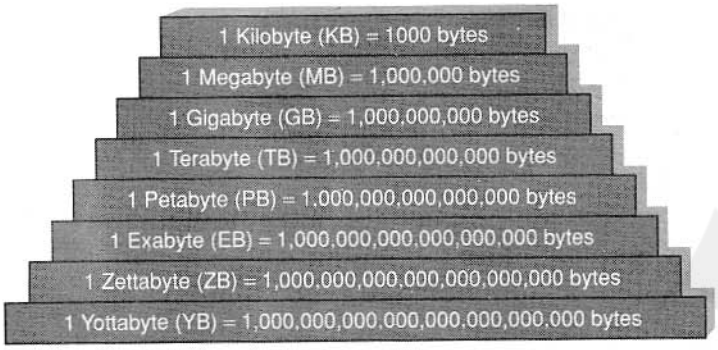


Figure 2.6 A mountain of data.

Figure 2.7 Sources of big data.

2. **External data sources:** Data residing outside an organization's firewall. It is as follows:
   - ***Public Web:*** Wikipedia, weather, regulatory, compliance, census, etc.
3. **Both (internal + external data sources)**
   - ***Sensor data:*** Car sensors, smart electric meters, office buildings, air conditioning units, refrigerators, and so on.
   - ***Machine log data:*** Event logs, application logs, Business process logs, audit logs, clickstream data, etc.
   - ***Social media:*** Twitter, blogs, Facebook, LinkedIn, YouTube, Instagram, etc.
   - ***Business apps:*** ERP, CRM, HR, Google Docs, and so on.
   - ***Media:*** Audio, Video, Image, Podcast, etc.
   - ***Docs:*** Comma separated value (CSV), Word Documents, PDF, XLS, PPT, and so on.

2.5.2 Velocity

We have moved from the days of batch processing (remember our payroll applications) to real-time processing.

Batch → Periodic → Near real time → Real-time processing

2.5.3 Variety

Variety deals with a wide range of data types and sources of data. We will study this under three categories: Structured data, semi-structured data and unstructured data.

1. **Structured data:** From traditional transaction processing systems and RDBMS, etc.
2. **Semi-structured data:** For example Hyper Text Markup Language (HTML), eXtensible Markup Language (XML).
3. **Unstructured data:** For example unstructured text documents, audios, videos, emails, photos, PDFs, social media, etc.

2.6 OTHER CHARACTERISTICS OF DATA WHICH ARE NOT DEFINITIONAL TRAITS OF BIG DATA

There are yet other characteristics of data which are not necessarily the definitional traits of big data. Few of these are listed as follows:

1. **Veracity and validity:** *Veracity* refers to biases, noise, and abnormality in data. The key question here is: "Is all the data that is being stored, mined, and analyzed meaningful and pertinent to the problem under consideration?" *Validity* refers to the accuracy and correctness of the data. Any data that is picked up for analysis needs to be accurate. It is not just true about big data alone.
2. **Volatility:** Volatility of data deals with, how long is the data valid? And how long should it be stored? There is some data that is required for long-term decisions and remains valid for longer periods of time. However, there are also pieces of data that quickly become obsolete minutes after their generation.
3. **Variability:** Data flows can be highly inconsistent with periodic peaks.

PICTURE THIS...

An online retailer announces the "big sale day" for a particular week. The retailer is likely to experience an upsurge in customer traffic to the website during this week. In the same way, he/she might experience a slump in his/her business immediately after the festival season. This reemphasizes the point that one might witness spikes in data at some point in time and at other times, the data flow can go flat.

2.7 WHY BIG DATA?

The more data we have for analysis, the greater will be the analytical accuracy and also the greater would be the confidence in our decisions based on these analytical findings. This will entail a greater positive impact in terms of enhancing operational efficiencies, reducing cost and time, and innovating on new products, new services, and optimizing existing services. Refer Figure 2.8.

More data → More accurate analysis → Greater confidence in decision making → Greater operational efficiencies, cost reduction, time reduction, new product development, and optimized offerings, etc.

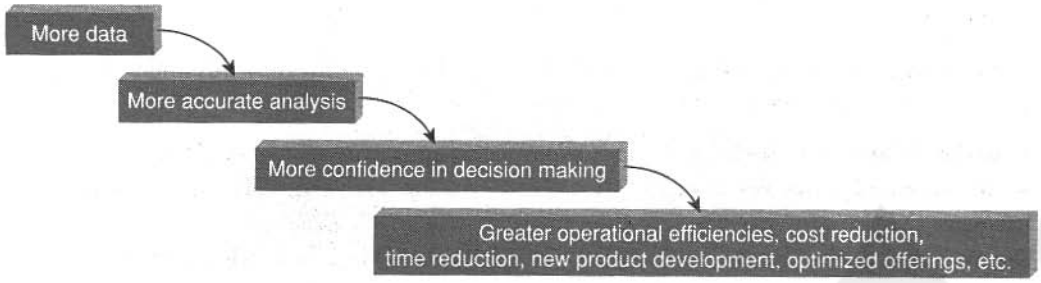


Figure 2.8 Why big data?

2.8 ARE WE JUST AN INFORMATION CONSUMER OR DO WE ALSO PRODUCE INFORMATION?

PICTURE THIS...

You have been invited to your friend's promotion party. You are happy and excited to join your friend at this important milestone in her career. You send in your confirmation through a text message. You get ready and leave for your friend's residence. On the way, you stop at a gas station to refuel. You pay using your credit card. You stop at an upmarket Archie's store to pick a good greeting card and a gift. You get the items billed at the Point of Sale system and pay cash at the counter. While at the party, you click photographs and post it on Facebook, Flickr, and the likes. Within minutes, you start to get likes and comments on your posts.

Mention the places in this scenario where data was generated:

1. Text message to send in the confirmation to attend the promotion bash.
2. Use of credit card to pay for gas/fuel at the gas station.
3. Point of Sale system at Archie's where your transaction gets recorded.
4. Photographs and posts on social networking sites.
5. Likes and comments to your post.

Likewise, there are several instances everyday where you generate data. Think about cases where you are a consumer of information.

2.9 TRADITIONAL BUSINESS INTELLIGENCE (BI) VERSUS BIG DATA

Let us take a sneak peek into some of the differences that one encounters dealing with traditional BI and big data.

1. In traditional BI environment, all the enterprise's data is housed in a central server whereas in a big data environment data resides in a distributed file system. The distributed file system scales by scaling in or out horizontally as compared to typical database server that scales vertically.
2. In traditional BI, data is generally analyzed in an offline mode whereas in big data, it is analyzed in both real time as well as in offline mode.

3. Traditional BI is about structured data and it is here that data is taken to processing functions (move data to code) whereas big data is about variety: Structured, semi-structured, and unstructured data and here the processing functions are taken to the data (move code to data).

2.10 A TYPICAL DATA WAREHOUSE ENVIRONMENT

Let us look at a typical Data Warehouse (DW) environment. Operational or transactional or day-to-day business data is gathered from Enterprise Resource Planning (ERP) systems, Customer Relationship Management (CRM), legacy systems, and several third party applications. The data from these sources may differ in format [data could have been housed in any RDBMS such as Oracle, MS SQL Server, DB2, MySQL, and Teradata, and so on or in spreadsheet (.xls, .xlsx, etc.) or .csv or txt]. Data may come from data sources located in the same geography or different geographies. This data is then integrated, cleaned up, transformed, and standardized through the process of Extraction, Transformation, and Loading (ETL). The transformed data is then loaded into the enterprise data warehouse (available at the enterprise level) or data marts (available at the business unit/ functional unit or business process level). A host of market leading business intelligence and analytics tools are then used to enable decision making from the use of ad-hoc queries, SQL, enterprise dashboards, data mining, etc. Refer Figure 2.9.

2.11 A TYPICAL HADOOP ENVIRONMENT

Let us now study the Hadoop environment. Is it very different from the data warehouse environment and where exactly is this difference?

As is fairly obvious from Figure 2.10, the data sources are quite disparate from web logs to images, audios, and videos to social media data to the various docs, pdfs, etc. Here the data in focus is not just the data within the company's firewall but also data residing outside the company's firewall. This data is placed in Hadoop Distributed File System (HDFS). If need be, this can be repopulated back to operational systems or fed to the enterprise data warehouse or data marts or Operational Data Store (ODS) to be picked for further processing and analysis.

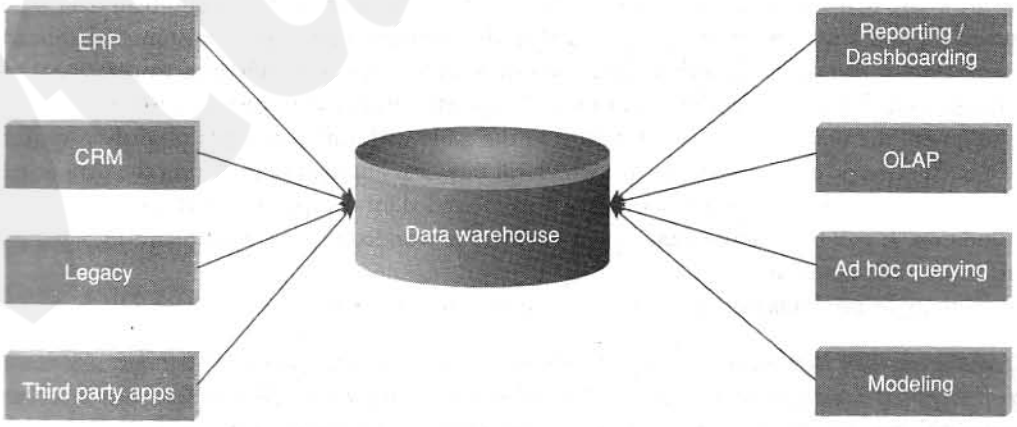


Figure 2.9 A typical data warehouse environment.

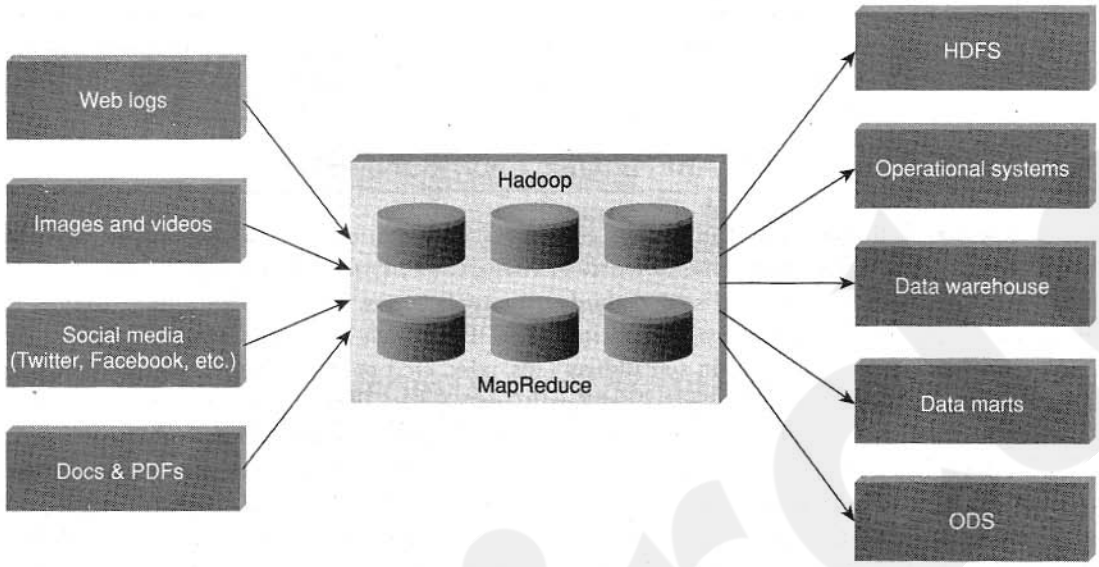


Figure 2.10 A typical Hadoop environment.

3.2 WHAT IS BIG DATA ANALYTICS?

Big Data Analytics is...

1. **Technology-enabled analytics:** Quite a few data analytics and visualization tools are available in the market today from leading vendors such as IBM, Tableau, SAS, R Analytics, Statistica, World Programming Systems (WPS), etc. to help process and analyze your big data.
2. About gaining a meaningful, deeper, and richer insight into your business to steer it in the right direction, understanding the customer's demographics to cross-sell and up-sell to them, better leveraging the services of your vendors and suppliers, etc.

   Author's experience: The other day I was pleasantly surprised to get a few recommendations via email from one of my frequently visited online retailers. They had recommended clothing line from my favorite brand and also the color suggested was one to my liking. How did they arrive at this? In the recent past, I had been buying clothing line of a particular brand and the color preference was pastel shades. They had it stored in their database and pulled it out while making recommendations to me.
3. About a competitive edge over your competitors by enabling you with findings that allow quicker and better decision-making.
4. A tight handshake between three communities: IT, business users, and data scientists.
   Refer Figure 3.3.
5. Working with datasets whose volume and variety exceed the current storage and processing capabilities and infrastructure of your enterprise.
6. About moving code to data. This makes perfect sense as the program for distributed processing is tiny (just a few KBs) compared to the data (Terabytes or Petabytes today and likely to be Exabytes or Zettabytes in the near future).

3.3 WHAT BIG DATA ANALYTICS ISN'T?

We have often asked participants of our learning programs as what comes to mind when you hear the term "Big Data." And we are not surprised by the answer... it is "Volume." But now that we have a clear understanding of big data, we know it isn't only about volume but the variety and velocity too are very important factors.

And before we think it is only used by huge online companies like a Google or Amazon, let us clear the myth. It is for any business and any industry that needs actionable insights out of their data (both internal and external).

3.4 WHY THIS SUDDEN HYPE AROUND BIG DATA ANALYTICS?

If we go by the industry buzz, every place there seems to be talk about big data and big data analytics. Why this sudden hype? Refer Figure 3.5.

Let us put it down to three foremost reasons:

1. Data is growing at a 40% compound annual rate, reaching nearly 45 ZB by 2020. In 2010, almost about 1.2 trillion Gigabyte of data was generated. This amount doubled to 2.4 trillion Gigabyte in 2012 and to about 5 trillion Gigabytes in the year 2014. The volume of business data worldwide is expected to double every 1.2 years. Wal-Mart, the world retailer, processes one million customer transactions per hour. 500 million "tweets" are posted by Twitter users every day. 2.7 billion "Likes" and comments are posted by Facebook users in a day. Every day 2.5 quintillion bytes of data is created, with 90% of the world's data created in the past 2 years alone.

   Source:

   (a) http://www.intel.com/content/www/us/en/communications/internet-minute-infographic.html

   (b) http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html
2. Cost per gigabyte of storage has hugely dropped.
3. There are an overwhelming number of user-friendly analytics tools available in the market today.

3.5 CLASSIFICATION OF ANALYTICS

There are basically two schools of thought:

1. Those that classify analytics into basic, operationalized, advanced, and monetized.
2. Those that classify analytics into analytics 1.0, analytics 2.0, and analytics 3.0.

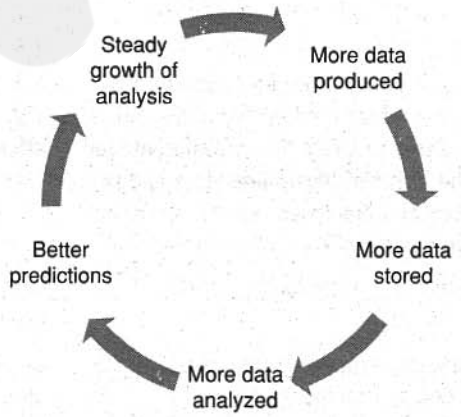


Figure 3.5 What big data entails?

3.5.1 First School of Thought

1. **Basic analytics:** This primarily is slicing and dicing of data to help with basic business insights. This is about reporting on historical data, basic visualization, etc.
2. **Operationalized analytics:** It is operationalized analytics if it gets woven into the enterprise's business processes.
3. **Advanced analytics:** This largely is about forecasting for the future by way of predictive and prescriptive modeling.
4. **Monetized analytics:** This is analytics in use to derive direct business revenue.

3.5.2 Second School of Thought

Let us take a closer look at analytics 1.0, analytics 2.0, and analytics 3.0. Refer Table 3.1.

Table 3.1 Analytics 1.0, 2.0, and 3.0

Analytics 1.0	Analytics 2.0	Analytics 3.0
Era: mid 1950s to 2009	**2005 to 2012**	**2012 to present**
Descriptive statistics (report on events, occurrences, etc. of the past)	Descriptive statistics + predictive statistics (use data from the past to make predictions for the future)	Descriptive + predictive + prescriptive statistics (use data from the past to make prophecies for the future and at the same time make recommendations to leverage the situation to one's advantage)
Key questions asked: What happened? Why did it happen?	Key questions asked: What will happen? Why will it happen?	Key questions asked: What will happen? When will it happen? Why will it happen? What should be the action taken to take advantage of what will happen?
Data from legacy systems, ERP, CRM, and 3rd party applications.	Big data	A blend of big data and data from legacy systems, ERP, CRM, and 3rd party applications.
Small and structured data sources. Data stored in enterprise data warehouses or data marts.	Big data is being taken up seriously. Data is mainly unstructured, arriving at a much higher pace. This fast flow of data entailed that the influx of big volume data had to be stored and processed rapidly, often on massive parallel servers running Hadoop.	A blend of big data and traditional analytics to yield insights and offerings with speed and impact.
Data was internally sourced.	Data was often externally sourced.	Data is both being internally and externally sourced.
Relational databases	Database appliances, Hadoop clusters, SQL to Hadoop environments, etc.	In memory analytics, in database processing, agile analytical methods, machine learning techniques, etc.

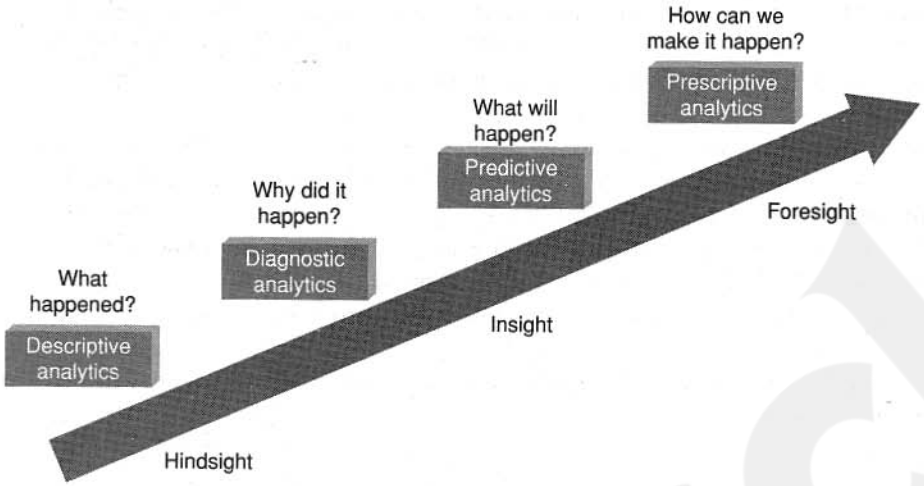


Figure 3.6 Analytics 1.0, 2.0, and 3.0.

Figure 3.6 shows the subtle growth of analytics from Descriptive → Diagnostic → Predictive → Prescriptive analytics.

3.6 GREATEST CHALLENGES THAT PREVENT BUSINESSES FROM CAPITALIZING ON BIG DATA

1. Obtaining executive sponsorships for investments in big data and its related activities (such as training, etc.).
2. Getting the business units to share information across organizational silos.
3. Finding the right skills (business analysts and data scientists) that can manage large amounts of structured, semi-structured, and unstructured data and create insights from it.
4. Determining the approach to scale rapidly and elastically. In other words, the need to address the storage and processing of large volume, velocity, and variety of big data.
5. Deciding whether to use structured or unstructured, internal or external data to make business decisions.
6. Choosing the optimal way to report findings and analysis of big data (visual presentation and analytics) for the presentations to make the most sense.
7. Determining what to do with the insights created from big data.

3.7 TOP CHALLENGES FACING BIG DATA

1. **Scale:** Storage (RDBMS (Relational Database Management System) or NoSQL (Not only SQL)) is one major concern that needs to be addressed to handle the need for scaling rapidly and elastically. The need of the hour is a storage that can best withstand the onslaught of large volume, velocity, and variety of big data? Should you scale vertically or should you scale horizontally?

2. **Security:** Most of the NoSQL big data platforms have poor security mechanisms (lack of proper authentication and authorization mechanisms) when it comes to safeguarding big data. A spot that cannot be ignored given that big data carries credit card information, personal information, and other sensitive data.
3. **Schema:** Rigid schemas have no place. We want the technology to be able to fit our big data and not the other way around. The need of the hour is dynamic schema. Static (pre-defined schemas) are passé.
4. **Continuous availability:** The big question here is how to provide 24/7 support because almost all RDBMS and NoSQL big data platforms have a certain amount of downtime built in.
5. **Consistency:** Should one opt for consistency or eventual consistency?
6. **Partition tolerant:** How to build partition tolerant systems that can take care of both hardware and software failures?
7. **Data quality:** How to maintain data quality – data accuracy, completeness, timeliness, etc.? Do we have appropriate metadata in place?

3.8 WHY IS BIG DATA ANALYTICS IMPORTANT?

Let us study the various approaches to analysis of data and what it leads to.

1. **Reactive – Business Intelligence:** What does Business Intelligence (BI) help us with? It allows the businesses to make faster and better decisions by providing the right information to the right person at the right time in the right format. It is about analysis of the past or historical data and then displaying the findings of the analysis or reports in the form of enterprise dashboards, alerts, notifications, etc. It has support for both pre-specified reports as well as ad hoc querying.
2. **Reactive – Big Data Analytics:** Here the analysis is done on huge datasets but the approach is still reactive as it is still based on static data.
3. **Proactive – Analytics:** This is to support futuristic decision making by the use of data mining, predictive modeling, text mining, and statistical analysis. This analysis is not on big data as it still uses the traditional database management practices on big data and therefore has severe limitations on the storage capacity and the processing capability.
4. **Proactive – Big Data Analytics:** This is sieving through terabytes, petabytes, exabytes of information to filter out the relevant data to analyze. This also includes high performance analytics to gain rapid insights from big data and the ability to solve complex problems using more data.

3.9 WHAT KIND OF TECHNOLOGIES ARE WE LOOKING TOWARD TO HELP MEET THE CHALLENGES POSED BY BIG DATA?

1. The first requirement is of cheap and abundant storage.
2. We need faster processors to help with quicker processing of big data.
3. Affordable open-source, distributed big data platforms, such as Hadoop.
4. Parallel processing, clustering, virtualization, large grid environments (to distribute processing to a number of machines), high connectivity, and high throughputs rather than low latency.
5. Cloud computing and other flexible resource allocation arrangements.

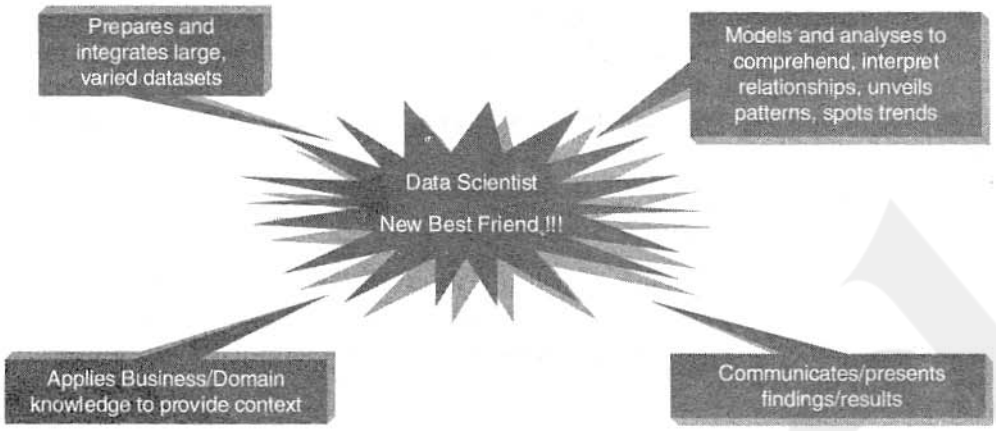


Figure 3.8 Data scientist: your new best friend!!!

2. **Analytical Techniques:** Depending on the business questions which we are trying to find answers to and the type of data available at hand, the data scientist employs a blend of analytical techniques to develop models and algorithms to understand the data, interpret relationships, spot trends, and unveil patterns.
3. **Business Analysts:** A data scientist is a business analyst who distinguishes cool facts from insights and is able to apply his business acumen and domain knowledge to see the results in the business context. He is a good presenter and communicator who is able to communicate the results of his findings in a language that is understood by the different business stakeholders.

3.12 TERMINOLOGIES USED IN BIG DATA ENVIRONMENTS

In order to get a good handle on the big data environment, let us get familiar with a few key terminologies in this arena.

3.12.1 In-Memory Analytics

Data access from non-volatile storage such as hard disk is a slow process. The more the data is required to be fetched from hard disk or secondary storage, the slower the process gets. One way to combat this challenge is to pre-process and store data (cubes, aggregate tables, query sets, etc.) so that the CPU has to fetch a small subset of records. But this requires thinking in advance as to what data will be required for analysis. If there is a need for different or more data, it is back to the initial process of pre-computing and storing data or fetching it from secondary storage.

This problem has been addressed using in-memory analytics. Here all the relevant data is stored in Random Access Memory (RAM) or primary storage thus eliminating the need to access the data from hard disk. The advantage is faster access, rapid deployment, better insights, and minimal IT involvement.

3.12.2 In-Database Processing

In-database processing is also called as *in-database analytics*. It works by fusing data warehouses with analytical systems. Typically the data from various enterprise On Line Transaction Processing (OLTP) systems after

cleaning up (de-duplication, scrubbing, etc.) through the process of ETL is stored in the Enterprise Data Warehouse (EDW) or data marts. The huge datasets are then exported to analytical programs for complex and extensive computations. With in-database processing, the database program itself can run the computations eliminating the need for export and thereby saving on time. Leading database vendors are offering this feature to large businesses.

3.12.3 Symmetric Multiprocessor System (SMP)

In SMP, there is a single common main memory that is shared by two or more identical processors. The processors have full access to all I/O devices and are controlled by a single operating system instance.

SMP are tightly coupled multiprocessor systems. Each processor has its own high-speed memory, called cache memory and are connected using a system bus. Refer Figure 3.9.

3.12.4 Massively Parallel Processing

Massive Parallel Processing (MPP) refers to the coordinated processing of programs by a number of processors working parallel. The processors, each have their own operating systems and dedicated memory. They work on different parts of the same program. The MPP processors communicate using some sort of messaging interface. The MPP systems are more difficult to program as the application must be divided in such a way that all the executing segments can communicate with each other. MPP is different from Symmetrically Multiprocessing (SMP) in that SMP works with the processors sharing the same operating system and same memory. SMP is also referred to as *tightly-coupled multiprocessing*.

3.12.5 Difference Between Parallel and Distributed Systems

The next two terms that we discuss are parallel and distributed systems.

As is evident from Figure 3.10, a parallel database system is a tightly coupled system. The processors co-operate for query processing. The user is unaware of the parallelism since he/she has no access to a specific

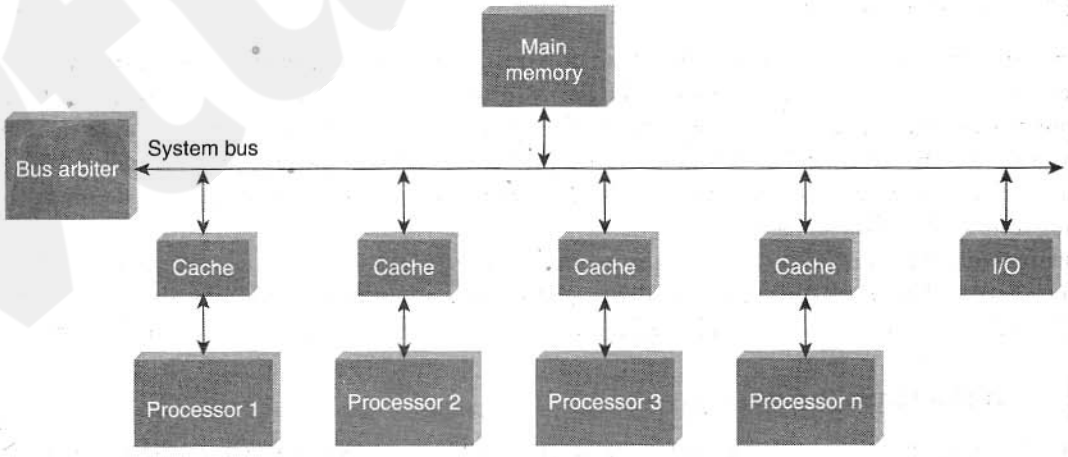


Figure 3.9 Symmetric Multiprocessor System.

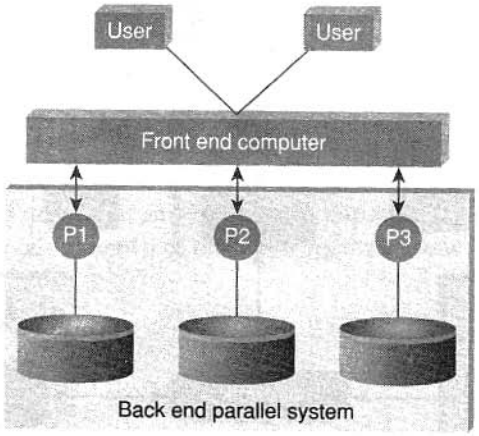


Figure 3.10 Parallel system.

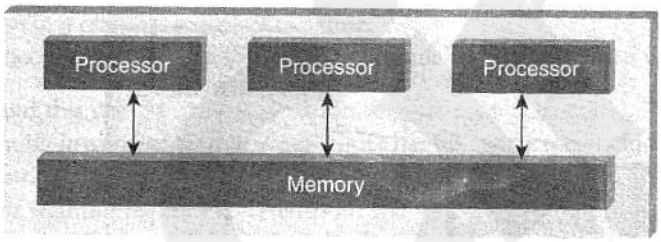


Figure 3.11 Parallel system.

processor of the system. Either the processors have access to a common memory (Refer Fig 3.11) or make use of message passing for communication.

Distributed database systems are known to be loosely coupled and are composed by individual machines. Refer Figure 3.12. Each of the machines can run their individual application and serve their own respective user. The data is usually distributed across several machines, thereby necessitating quite a number of machines to be accessed to answer a user query. Refer Figure 3.13.

3.12.6 Shared Nothing Architecture

Let us look at the three most common types of architecture for multiprocessor high transaction rate systems. They are:

1. Shared Memory (SM).
2. Shared Disk (SD).
3. Shared Nothing (SN).

In shared memory architecture, a common central memory is shared by multiple processors. In shared disk architecture, multiple processors share a common collection of disks while having their own private memory. In shared nothing architecture, neither memory nor disk is shared among multiple processors.

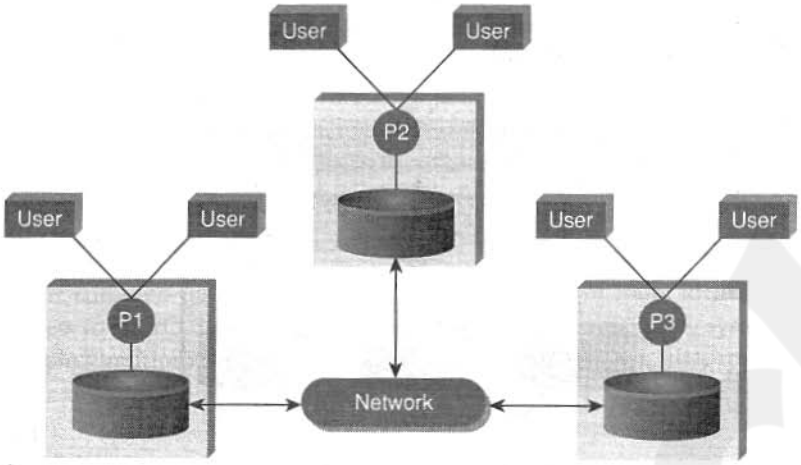


Figure 3.12 Distributed system.

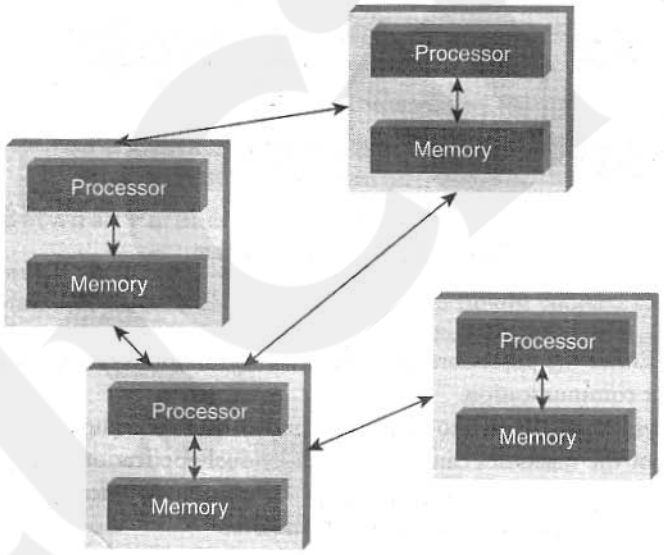


Figure 3.13 Distributed system.

3.12.6.1 Advantages of a "Shared Nothing Architecture"

1. **Fault Isolation:** A "Shared Nothing Architecture" provides the benefit of isolating fault. A fault in a single node is contained and confined to that node exclusively and exposed only through messages (or lack of it).
2. **Scalability:** Assume that the disk is a shared resource. It implies that the controller and the disk bandwidth are also shared. Synchronization will have to be implemented to maintain a consistent shared state. This would mean that different nodes will have to take turns to access the critical data. This

imposes a limit on how many nodes can be added to the distributed shared disk system, thus compromising on scalability.

3.12.7 CAP Theorem Explained

The CAP theorem is also called the *Brewer's Theorem*. It states that in a distributed computing environment (a collection of interconnected nodes that share data), it is impossible to provide the following guarantees. Refer Figure 3.14. At best you can have two of the following three – one must be sacrificed.

1. Consistency
2. Availability
3. Partition tolerance

3.12.7.1 CAP Theorem

Let us spend some time understanding the earlier mentioned terms.

1. Consistency implies that every read fetches the last write.
2. Availability implies that reads and writes always succeed. In other words, each non-failing node will return a response in a reasonable amount of time.
3. Partition tolerance implies that the system will continue to function when network partition occurs.

Let us try to understand this using a real-life situation.

You work for a training institute, "XYZ." The institute has 50 instructors including you. All of you report to a training coordinator. At the end of the month, all the instructors together with the training coordinator peruse through the training requests received from the various corporate houses and prepare a training schedule for each instructor. These training schedules (one for each instructor) are shared with "Amey," the office administrator. Each morning, you either call the office helpdesk (essentially Amey's desk) or check in-person with Amey for your schedule for the day. In case a training request has been cancelled or updated (updates can be in the form of change in course, change in duration, change of the training timings, etc.), Amey is informed of the updates and the schedules are subsequently updated by him.

Things were good until now. Few corporate houses were your clients and the schedules of each instructor could be smoothly managed without any major hiccups. But your training institute has been implementing promotion campaigns to expand the business. As a result of advertising in the media and word of mouth publicity by your existing clients, you suddenly see an upsurge in training requests from existing and new clients. In consequence of that, more instructors have been recruited. Few trainers/consultants have also been roped in from other training institutes to help tackle the load.

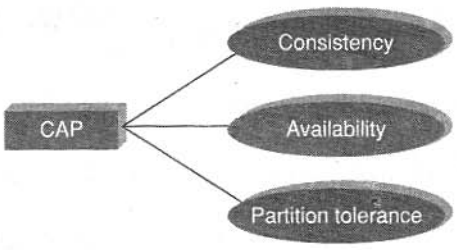


Figure 3.14 Brewer's CAP.

Now when you go to Amey to check your schedule or call in at the helpdesk, you are prepared for a wait in the queue. Looking at the current state of affairs, the training coordinator decides to recruit an additional office administrator "Joey." The helpdesk number will remain the same and will be shared by both the office administrators.

This arrangement works well for a couple of days. Then one day…

You: Hey Amey!

Amey: Hi! How can I help?

You: I think I am scheduled to anchor a training at 3:00 pm today. Can I please have the details?

Amey: Sure! Just a minute.

Amey browses through the file where he maintains the schedules. He does not see a training scheduled against your name at 3:00 pm today and responds back, "You do not have any training to conduct at 3:00 pm."

You: How is that possible? The training coordinator called up yesterday evening to inform of the same and said he has updated the office administrators of the same.

Amey: Oh! Did he say which office administrator? It could have been Joey. Please check with Joey.

Amey: Hey Joey! Please check the schedule for Paul here… Do you see something scheduled at 3:00 pm today?

Joey: Sure enough! He is anchoring the training for client "Z" today at 3:00 pm.

A clear case of inconsistent system!!! The updates in the schedule were shared by the training coordinator with Joey and you were checking for your schedule with Amey.

You share this incident with the training coordinator and that gets him thinking. The issue has to be addressed immediately otherwise it will be difficult to avoid a chaotic situation. He comes up with a plan and shares it with both the office administrators the following day.

Training Coordinator: Folks, each time that either an instructor or me calls any one of you to update a schedule, make sure that both of you update it in your respective files. This way the instructor will always get the most recent and consistent information irrespective of whom amongst the two of you he/she speaks to.

Joey: But that could mean a delay in answering either a phone call or sharing the schedule with the instructor waiting in queue.

Training Coordinator: Yes, I understand. But there is no way that we can give incorrect information.

Amey: There is this other problem as well. Suppose one of us is on leave on a particular day. That would mean that we cannot take any update related calls as we will not be able to simultaneously update both the files (my file and Joey's).

Training Coordinator: Well, good point! *That's the availability problem!!!* But I have thought about that as well. Here is the plan:

1. If one of you receives the update call (any updates to any schedule), ensure that you inform the other person if he is available.
2. In case the other person is not available, ensure that you inform him of all the updates to all schedules via email. It is a must!!!
3. When the other person resumes duty, the first thing he will do is update his file with all the updates to all schedules that he has received via email.

Wow!!! That is sure a Consistent and Available system!!!

Looks like everything is in control. Wait a minute! There is a tiff that has taken place between the office administrators. The two are pretty much available but are not talking to each other which, in other words, means that the updates are not flowing from one to the other. *We have to be partition tolerant!!!* As a training coordinator, you instruct them saying that none of you are taking any calls requesting for schedules or updates to schedules till you patch up. This implies that the system is partition tolerant but not available at that time.

In summary, one can at most decide to go with two of the three.

1. **Consistent:** The instructors or the training coordinator, once they have updated information with you, will always get the most updated information when they call subsequently.
2. **Availability:** The instructors or the training coordinators will always get the schedule if any or both of the office administrators have reported to work.
3. **Partition Tolerance:** Work will go on as usual even if there is communication loss between the office administrators owing to a spat or a tiff!

When to choose consistency over availability and vice-versa…

1. Choose availability over consistency when your business requirements allow some flexibility around when the data in the system synchronizes.
2. Choose consistency over availability when your business requirements demand atomic reads and writes.

Examples of databases that follow one of the possible three combinations

1. Availability and Partition Tolerance (AP)
2. Consistency and Partition Tolerance (CP)
3. Consistency and Availability (CA)

Refer Figure 3.15 to get a glimpse of databases that adhere to two of the three characteristics of CAP theorem.

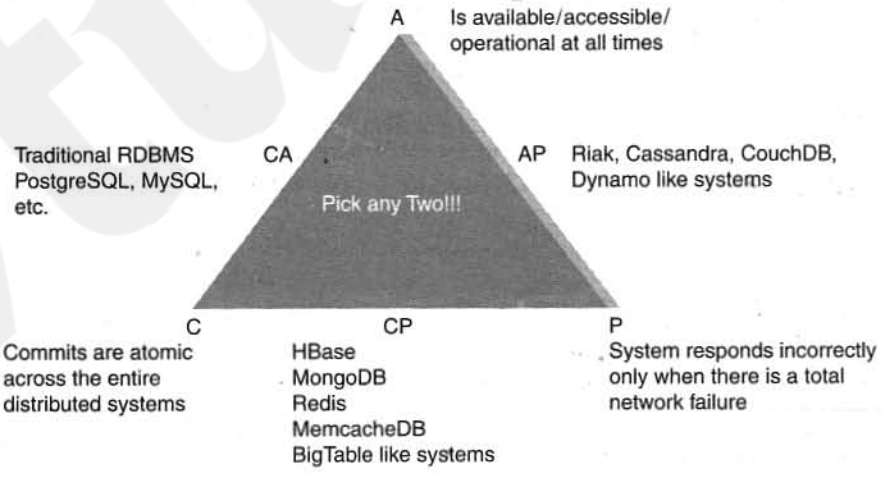


Figure 3.15 Databases and CAP.

4.1 NoSQL (NOT ONLY SQL)

The term NoSQL was first coined by Carlo Strozzi in 1998 to name his lightweight, open-source, relational database that did not expose the standard SQL interface. Johan Oskarsson, who was then a developer at last. fm, in 2009 reintroduced the term NoSQL at an event called to discuss open-source distributed network. The #NoSQL was coined by Eric Evans and few other database people at the event found it suitable to describe these non-relational databases.

Few features of NoSQL databases are as follows:

1. They are open source.
2. They are non-relational.
3. They are distributed.
4. They are schema-less.
5. They are cluster friendly.
6. They are born out of 21st century web applications.

4.1.1 Where is it Used?

NoSQL databases are widely used in big data and other real-time web applications. Refer Figure 4.1. NoSQL databases is used to stock log data which can then be pulled for analysis. Likewise it is used to store social media data and all such data which cannot be stored and analyzed comfortably in RDBMS.

4.1.2 What is it?

NoSQL stands for Not Only SQL. These are non-relational, open source, distributed databases. They are hugely popular today owing to their ability to scale out or scale horizontally and the adeptness at dealing with a rich variety of data: structured, semi-structured and unstructured data. Refer Figure 4.2 for additional features of NoSQL. NoSQL databases.

1. **Are non-relational:** They do not adhere to relational data model. In fact, they are either key–value pairs or document-oriented or column-oriented or graph-based databases.

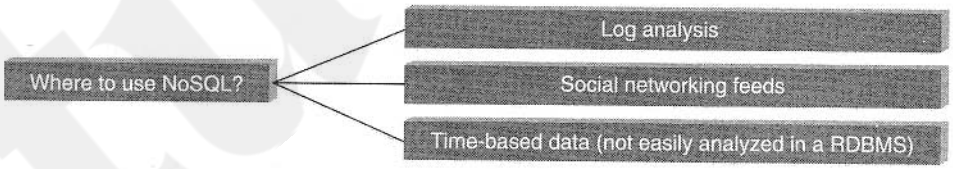


Figure 4.1 Where to use NoSQL?

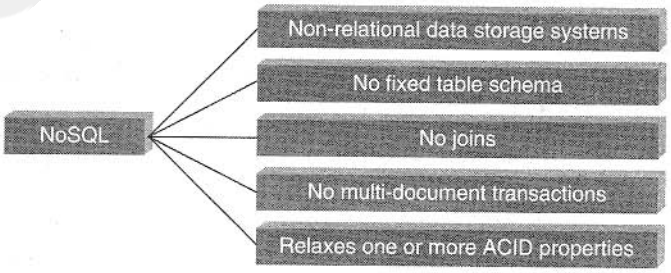


Figure 4.2 What is NoSQL?

2. **Are distributed:** They are distributed meaning the data is distributed across several nodes in a cluster constituted of low-cost commodity hardware.
3. **Offer no support for ACID properties (Atomicity, Consistency, Isolation, and Durability):** They do not offer support for ACID properties of transactions. On the contrary, they have adherence to Brewer's CAP (Consistency, Availability, and Partition tolerance) theorem and are often seen compromising on consistency in favor of availability and partition tolerance.
4. **Provide no fixed table schema:** NoSQL databases are becoming increasing popular owing to their support for flexibility to the schema. They do not mandate for the data to strictly adhere to any schema structure at the time of storage.

4.1.3 Types of NoSQL Databases

We have already stated that NoSQL databases are non-relational. They can be broadly classified into the following:

1. Key–value or the big hash table.
2. Schema-less.

Refer Figure 4.3. Let us take a closer look at key–value and few other types of schema-less databases:

1. **Key–value:** It maintains a big hash table of keys and values. For example, Dynamo, Redis, Riak, etc.
   Sample Key-Value Pair in Key-Value Database

Key	Value
First Name	Simmonds
Last Name	David

2. **Document:** It maintains data in collections constituted of documents. For example, MongoDB, Apache CouchDB, Couchbase, MarkLogic, etc.
   Sample Document in Document Database

```
{
"Book Name":    "Fundamentals of Business Analytics",
"Publisher":    "Wiley India",
"Year of Publication":    "2011"
}
```

3. **Column:** Each storage block has data from only one column. For example: Cassandra, HBase, etc.

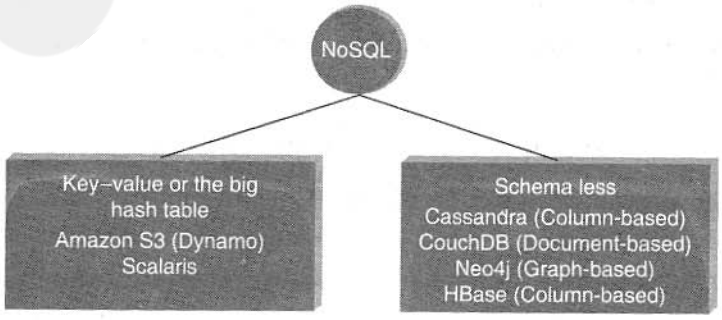
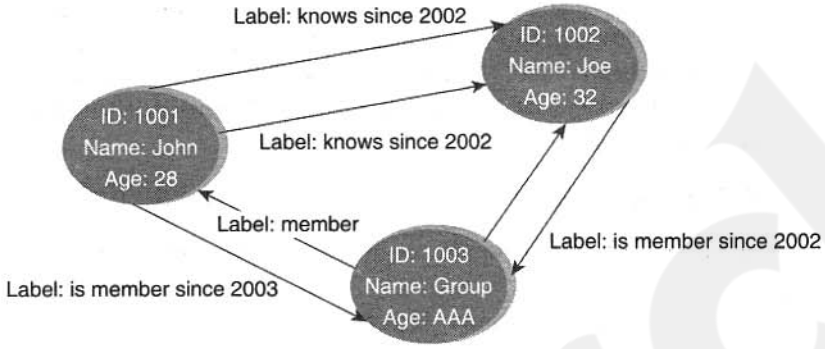


Figure 4.3 Types of NoSQL databases.

4. **Graph:** They are also called network database. A graph stores data in nodes. For example, Neo4j, HyperGraphDB, etc.
   Sample Graph in Graph Database



Refer Table 4.1 for popular schema-less databases.

4.1.4 Why NoSQL?

1. It has scale out architecture instead of the monolithic architecture of relational databases.
2. It can house large volumes of structured, semi-structured, and unstructured data.
3. **Dynamic schema:** NoSQL database allows insertion of data without a pre-defined schema. In other words, it facilitates application changes in real time, which thus supports faster development, easy code integration, and requires less database administration.
4. **Auto-sharding:** It automatically spreads data across an arbitrary number of servers. The application in question is more often not even aware of the composition of the server pool. It balances the load of data and query on the available servers; and if and when a server goes down, it is quickly replaced without any major activity disruptions.
5. **Replication:** It offers good support for replication which in turn guarantees high availability, fault tolerance, and disaster recovery.

4.1.5 Advantages of NoSQL

Let us enumerate the advantages of NoSQL. Refer Figure 4.4.

1. **Can easily scale up and down:** NoSQL database supports scaling rapidly and elastically and even allows to scale to the cloud.

Table 4.1 Popular schema-less databases

Key–Value Data Store	Column-Oriented Data Store	Document Data Store	Graph Data Store
• Riak	• Cassandra	• MongoDB	• InfiniteGraph
• Redis	• HBase	• CouchDB	• Neo4j
• Membase	• HyperTable	• RavenDB	• AllegroGraph

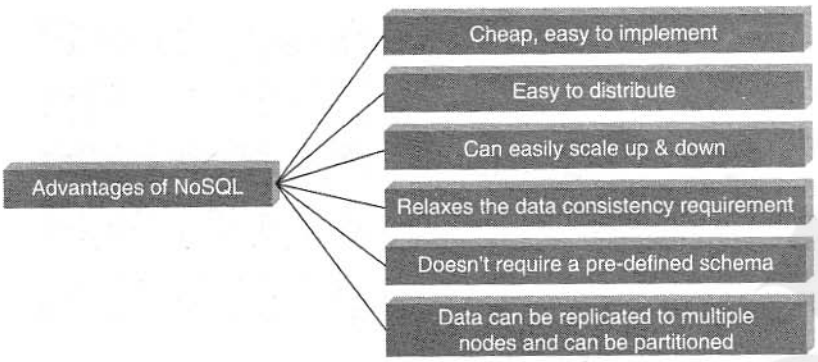


Figure 4.4 Advantages of NoSQL.

(a) ***Cluster scale:*** It allows distribution of database across 100+ nodes often in multiple data centers.
(b) ***Performance scale:*** It sustains over 100,000+ database reads and writes per second.
(c) ***Data scale:*** It supports housing of 1 billion+ documents in the database.

2. **Doesn't require a pre-defined schema:** NoSQL does not require any adherence to pre-defined schema. It is pretty flexible. For example, if we look at MongoDB, the documents (equivalent of records in RDBMS) in a collection (equivalent of table in RDBMS) can have different sets of key–value pairs.

   {_id: 101,"BookName": "Fundamentals of Business Analytics", "AuthorName": "Seema Acharya", "Publisher": "Wiley India"}
   {_id:102, "BookName":"Big Data and Analytics"}

3. **Cheap, easy to implement:** Deploying NoSQL properly allows for all of the benefits of scale, high availability, fault tolerance, etc. while also lowering operational costs.
4. **Relaxes the data consistency requirement:** NoSQL databases have adherence to CAP theorem (Consistency, Availability, and Partition tolerance). Most of the NoSQL databases compromise on consistency in favor of availability and partition tolerance. However, they do go for eventual consistency.
5. **Data can be replicated to multiple nodes and can be partitioned:** There are two terms that we will discuss here:

   (a) ***Sharding:*** Sharding is when different pieces of data are distributed across multiple servers. NoSQL databases support auto-sharding; this means that they can natively and automatically spread data across an arbitrary number of servers, without requiring the application to even be aware of the composition of the server pool. Servers can be added or removed from the data layer without application downtime. This would mean that data and query load are automatically balanced across servers, and when a server goes down, it can be quickly and transparently replaced with no application disruption.
   (b) ***Replication:*** Replication is when multiple copies of data are stored across the cluster and even across data centers. This promises high availability and fault tolerance.

4.1.6 What We Miss With NoSQL?

With NoSQL around, we have been able to counter the problem of scale (NoSQL scales out). There is also the flexibility with respect to schema design. However there are few features of conventional RDBMS that are greatly missed. Refer Figure 4.5.

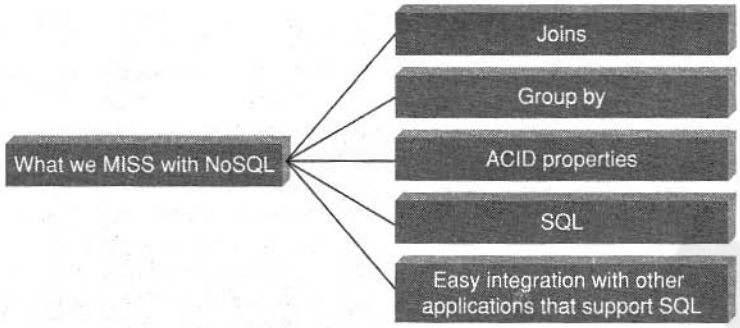


Figure 4.5 What we miss with NoSQL?

NoSQL does not support joins. However, it compensates for it by allowing embedded documents as in MongoDB. It does not have provision for ACID properties of transactions. However, it obeys the Eric Brewer's CAP theorem. NoSQL does not have a standard SQL interface but NoSQL databases such as MongoDB and Cassandra have their own rich query language [MongoDB query language and Cassandra query language (CQL)] to compensate for the lack of it. One thing which is dearly missed is the easy integration with other applications that support SQL.

4.1.7 Use of NoSQL in Industry

NoSQL is being put to use in varied industries. They are used to support analysis for applications such as web user data analysis, log analysis, sensor feed analysis, making recommendations for upsell and cross-sell, etc. Refer Figure 4.6.

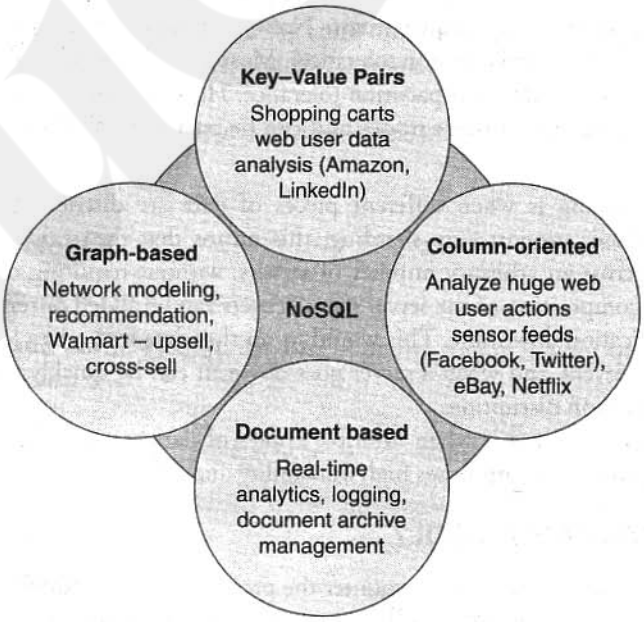


Figure 4.6 Use of NoSQL in industry.

4.1.8 NoSQL Vendors

Refer Table 4.2 for few popular NoSQL vendors.

Table 4.2 Few popular NoSQL vendors

Company	Product	Most Widely Used by
Amazon	DynamoDB	LinkedIn, Mozilla
Facebook	Cassandra	Netflix, Twitter, eBay
Google	BigTable	Adobe Photoshop

4.1.9 SQL versus NoSQL

Refer Table 4.3 for few salient differences between SQL and NoSQL.

Table 4.3 SQL versus NoSQL

SQL	NoSQL
Relational database	Non-relational, distributed database
Relational model	Model-less approach
Pre-defined schema	Dynamic schema for unstructured data
Table based databases	Document-based or graph-based or wide column store or key–value pairs databases
Vertically scalable (by increasing system resources)	Horizontally scalable (by creating a cluster of commodity machines)
Uses SQL	Uses UnQL (Unstructured Query Language)
Not preferred for large datasets	Largely preferred for large datasets
Not a best fit for hierarchical data	Best fit for hierarchical storage as it follows the key–value pair of storing data similar to JSON (Java Script Object Notation)
Emphasis on ACID properties	Follows Brewer's CAP theorem
Excellent support from vendors	Relies heavily on community support
Supports complex querying and data keeping needs	Does not have good support for complex querying
Can be configured for strong consistency	Few support strong consistency (e.g., MongoDB), some others can be configured for eventual consistency (e.g., Cassandra)
Examples: Oracle, DB2, MySQL, MS SQL, PostgreSQL, etc.	Examples: MongoDB, HBase, Cassandra, Redis, Neo4j, CouchDB, Couchbase, Riak, etc.

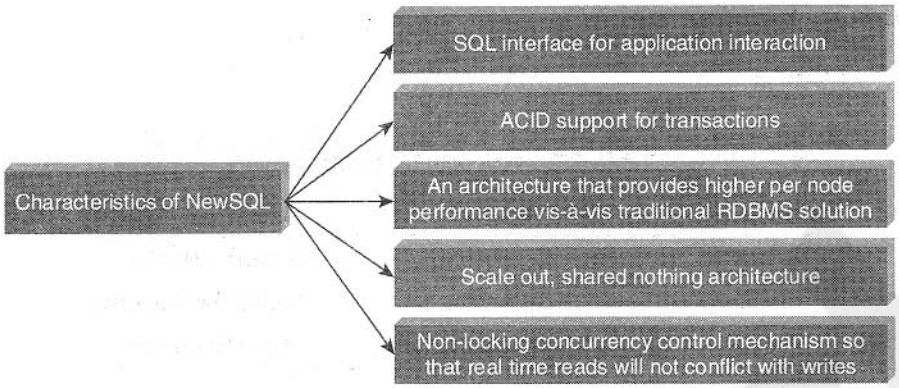


Figure 4.7 Characteristics of NewSQL.

4.1.10 NewSQL

There is yet another new term doing the rounds – "NewSQL". So, what is NewSQL and how is it different from SQL and NoSQL?

What is that we love about NoSQL and is not there with our traditional RDBMS and what is that we love about SQL that NoSQL does not have support for? You guessed it right!!! We need a database that has the same scalable performance of NoSQL systems for On Line Transaction Processing (OLTP) while still maintaining the ACID guarantees of a traditional database. This new modern RDBMS is called NewSQL. It supports relational data model and uses SQL as their primary interface.

4.1.10.1 Characteristics of NewSQL

Refer Figure 4.7 to learn about the characteristics of NewSQL. NewSQL is based on the shared nothing architecture with a SQL interface for application interaction.

4.1.11 Comparison of SQL, NoSQL, and NewSQL

Refer Table 4.4 for a comparative study of SQL, NoSQL and NewSQL.

Table 4.4 Comparative study of SQL, NoSQL and NewSQL

	SQL	NoSQL	NewSQL
Adherence to ACID properties	Yes	No	Yes
OLTP/OLAP	Yes	No	Yes
Schema rigidity Adherence to data model	Yes Adherence to relational model	No	Maybe
Data Format Flexibility	No	Yes	Maybe
Scalability	Scale up Vertical Scaling	Scale out Horizontal Scaling	Scale out
Distributed Computing	Yes	Yes	Yes
Community Support	Huge	Growing	Slowly growing

4.2 HADOOP

Hadoop is an open-source project of the Apache foundation. It is a framework written in Java, originally developed by Doug Cutting in 2005 who named it after his son's toy elephant. He was working with Yahoo then. It was created to support distribution for "Nutch", the text search engine. Hadoop uses Google's MapReduce and Google File System technologies as its foundation. Hadoop is now a core part of the computing infrastructure for companies such as Yahoo, Facebook, LinkedIn, Twitter, etc. Refer Figure 4.8.

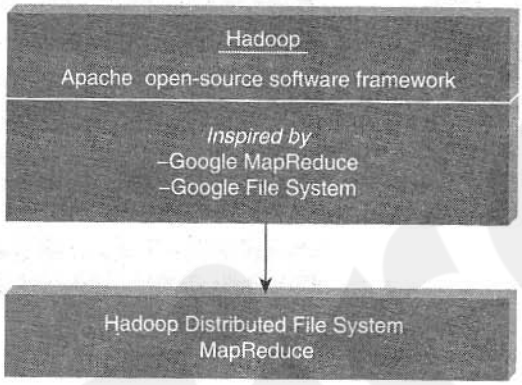


Figure 4.8 Hadoop.

4.2.1 Features of Hadoop

Let us cite a few features of Hadoop:

1. It is optimized to handle massive quantities of structured, semi-structured, and unstructured data, using commodity hardware, that is, relatively inexpensive computers.
2. Hadoop has a shared nothing architecture.
3. It replicates its data across multiple computers so that if one goes down, the data can still be processed from another machine that stores its replica.
4. Hadoop is for high throughput rather than low latency. It is a batch operation handling massive quantities of data; therefore the response time is not immediate.
5. It complements On-Line Transaction Processing (OLTP) and On-Line Analytical Processing (OLAP). However, it is not a replacement for a relational database management system.
6. It is NOT good when work cannot be parallelized or when there are dependencies within the data.
7. It is NOT good for processing small files. It works best with huge data files and datasets.

4.2.2 Key Advantages of Hadoop

Refer Figure 4.9 for a quick look at the key advantages of Hadoop. Some of them are as follows:

1. **Stores data in its native format:** Hadoop's data storage framework (HDFS – Hadoop Distributed File System) can store data in its native format. There is no structure that is imposed while keying in data or storing data. HDFS is pretty much schema-less. It is only later when the data needs to be processed that structure is imposed on the raw data.
2. **Scalable:** Hadoop can store and distribute very large datasets (involving thousands of terabytes of data) across hundreds of inexpensive servers that operate in parallel.

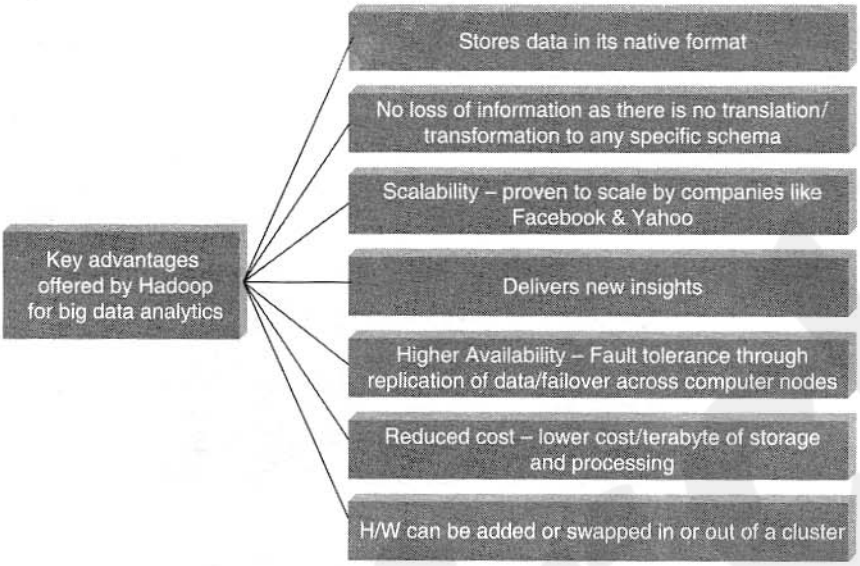


Figure 4.9 Key advantages of Hadoop.

3. **Cost-effective:** Owing to its scale-out architecture, Hadoop has a much reduced cost/terabyte of storage and processing.
4. **Resilient to failure:** Hadoop is fault-tolerant. It practices replication of data diligently which means whenever data is sent to any node, the same data also gets replicated to other nodes in the cluster, thereby ensuring that in the event of a node failure, there will always be another copy of data available for use.
5. **Flexibility:** One of the key advantages of Hadoop is its ability to work with all kinds of data: structured, semi-structured, and unstructured data. It can help derive meaningful business insights from email conversations, social media data, click-stream data, etc. It can be put to several purposes such as log analysis, data mining, recommendation systems, market campaign analysis, etc.
6. **Fast:** Processing is extremely fast in Hadoop as compared to other conventional systems owing to the "move code to data" paradigm.

   Hadoop has a shared-nothing architecture.

4.2.3 Versions of Hadoop

There are two versions of Hadoop available:

1. Hadoop 1.0
2. Hadoop 2.0

Let us take a look at the features of both. Refer Figure 4.10.

4.2.3.1 Hadoop 1.0

It has two main parts:

1. **Data storage framework:** It is a general-purpose file system called Hadoop Distributed File System (HDFS). HDFS is schema-less. It simply stores data files. These data files can be in just about any

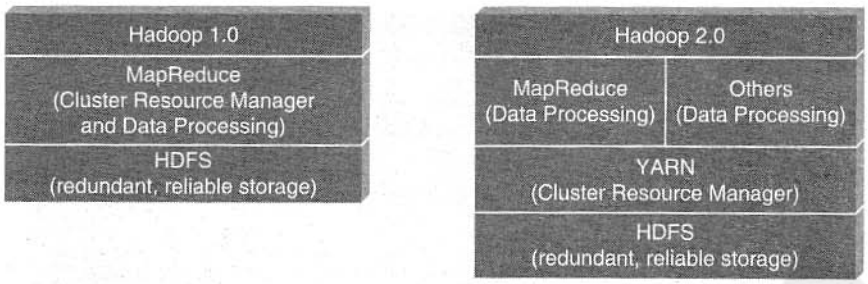


Figure 4.10 Versions of Hadoop.

format. The idea is to store files as close to their original form as possible. This is turn provides the business units and the organization the much needed flexibility and agility without being overly worried by what it can implement.

2. **Data processing framework:** This is a simple functional programming model initially popularized by Google as MapReduce. It essentially uses two functions: the MAP and the REDUCE functions to process data. The "Mappers" take in a set of key–value pairs and generate intermediate data (which is another list of key–value pairs). The "Reducers" then act on this input to produce the output data. The two functions seemingly work in isolation from one another, thus enabling the processing to be highly distributed in a highly-parallel, fault-tolerant, and scalable way.

There were, however, a few limitations of Hadoop 1.0. They are as follows:

1. The first limitation was the requirement for MapReduce programming expertise along with proficiency required in other programming languages, notably Java.
2. It supported only batch processing which although is suitable for tasks such as log analysis, large-scale data mining projects but pretty much unsuitable for other kinds of projects.
3. One major limitation was that Hadoop 1.0 was tightly computationally coupled with MapReduce, which meant that the established data management vendors were left with two options: Either rewrite their functionality in MapReduce so that it could be executed in Hadoop or extract the data from HDFS and process it outside of Hadoop. None of the options were viable as it led to process inefficiencies caused by the data being moved in and out of the Hadoop cluster.

Let us look at whether these limitations have been wholly or in parts resolved by Hadoop 2.0.

4.2.3.2 Hadoop 2.0

In Hadoop 2.0, HDFS continues to be the data storage framework. However, a new and separate resource management framework called **Y**et **A**nother **R**esource **N**egotiator (YARN) has been added. Any application capable of dividing itself into parallel tasks is supported by YARN. YARN coordinates the allocation of subtasks of the submitted application, thereby further enhancing the flexibility, scalability, and efficiency of the applications. It works by having an ApplicationMaster in place of the erstwhile JobTracker, running applications on resources governed by a new NodeManager (in place of the erstwhile TaskTracker). ApplicationMaster is able to run any application and not just MapReduce.

This, in other words, means that the MapReduce Programming expertise is no longer required. Furthermore, it not only supports batch processing but also real-time processing. MapReduce is no longer the only data processing option; other alternative data processing functions such as data standardization, master data management can now be performed natively in HDFS.

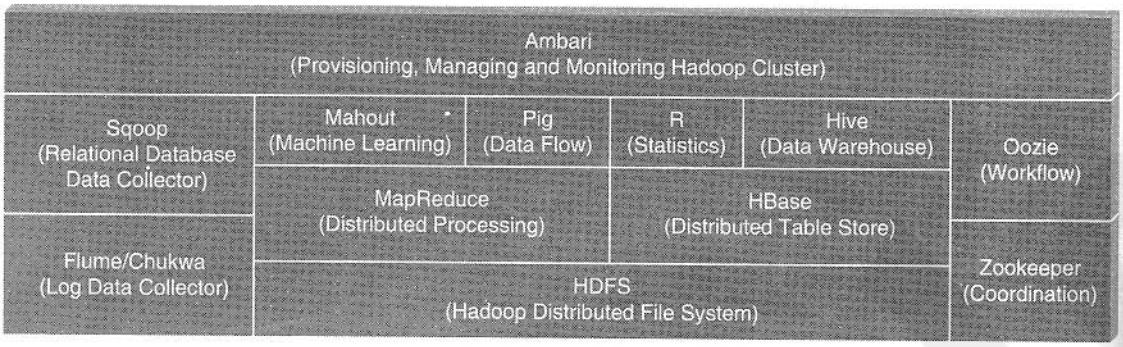


Figure 4.11 Hadoop ecosystem.

4.2.4 Overview of Hadoop Ecosystems

The components of the Hadoop ecosystem are shown in Figure 4.11.

There are components available in the Hadoop ecosystem for data ingestion, processing, and analysis.

Data Ingestion → Data Processing → Data Analysis

Components that help with Data Ingestion are:

1. Sqoop
2. Flume

Components that help with Data Processing are:

1. MapReduce
2. Spark

Components that help with Data Analysis are:

1. Pig
2. Hive
3. Impala

HDFS

It is the distributed storage unit of Hadoop. It provides streaming access to file system data as well as file permissions and authentication. It is based on GFS (Google File System). It is used to scale a single cluster node to hundreds and thousands of nodes. It handles large datasets running on commodity hardware. HDFS is highly fault-tolerant. It stores files across multiple machines. These files are stored in redundant fashion to allow for data recovery in case of failure.

PICTURE THIS...

An e-commerce website stores millions of customers' data in a distributed manner. Data has been collected over 4–5 years. It then runs batch analytics on the archived data to analyze customer's behavior, buying patterns, their preferences, their requirements, etc. This helps to understand which products are purchased by customers in which months, etc.

HBase

It stores data in HDFS. It is the first non-batch component of the Hadoop Ecosystem. It is a database on top of HDFS. It provides a quick random access to the stored data. It has very low latency compared to HDFS. It is a NoSQL database, is non-relational and is a column-oriented database. A table can have thousands of columns. A table can have multiple rows. Each row can have several column families. Each column family can have several columns. Each column can have several key values. It is based on Google BigTable. This is widely used by Facebook, Twitter, Yahoo, etc.

PICTURE THIS...

The same e-commerce website as in the HDFS case above also stores millions of product data. To search for a product among millions of products and to produce the result immediately (or you can say in real time), it needs to optimize the request and search process. HBase supports real-time analytics.

Given the huge velocity of data, they opted for HBase over HDFS, as HDFS does not support real-time writes. The results were overwhelming; it reduced the query time from 3 days to 3 minutes.

Difference between HBase and Hadoop/HDFS

1. HDFS is the file system whereas HBase is a Hadoop database. It is like NTFS and MySQL.
2. HDFS is WORM (Write once and read multiple times or many times). Latest versions support appending of data but this feature is rarely used. However, HBase supports real-time random read and write.
3. HDFS is based on Google File System (GFS) whereas HBase is based on Google Big Table.
4. HDFS supports only full table scan or partition table scan. Hbase supports random small range scan or table scan.
5. Performance of Hive on HDFS is relatively very good but for HBase it becomes 4–5 times slower.
6. The access to data is via MapReduce job only in HDFS whereas in HBase the access is via Java APIs, Rest, Avro, Thrift APIs.
7. HDFS does not support dynamic storage owing to its rigid structure whereas HBase supports dynamic storage.
8. HDFS has high latency operations whereas HBase has low latency operations.
9. HDFS is most suitable for batch analytics whereas HBase is for real-time analytics.

Hadoop Ecosystem Components for Data Ingestion

1. **Sqoop:** Sqoop stands for SQL to Hadoop. Its main functions are
   a) Importing data from RDBMS such as MySQL, Oracle, DB2, etc. to Hadoop file system (HDFS, HBase, Hive).
   b) Exporting data from Hadoop File system (HDFS, HBase, Hive) to RDBMS (MySQL, Oracle, DB2).

   Uses of Sqoop

   a) It has a connector-based architecture to allow plug-ins to connect to external systems such as MySQL, Oracle, DB2, etc.

b) It can provision the data from external system on to HDFS and populate tables in Hive and HBase.
c) It integrates with Oozie allowing you to schedule and automate import and export tasks.

2. **Flume:** Flume is an important log aggregator (aggregates logs from different machines and places them in HDFS) component in the Hadoop ecosystem. Flume has been developed by Cloudera. It is designed for high volume ingestion of event-based data into Hadoop. The default destination in Flume (called as sink in flume parlance) is HDFS. However it can also write to HBase or Solr.

PICTURE THIS...

There is a bank of web servers. Flume moves log events from those files into new aggregated files in HDFS for processing.

Hadoop Ecosystem Components for Data Processing

1. **MapReduce:** It is a programing paradigm that allows distributed and parallel processing of huge datasets. It is based on Google MapReduce. Google released a paper on MapReduce programming paradigm in 2004 and that became the genesis of Hadoop processing model. The MapReduce framework gets the input data from HDFS. There are two main phases: Map phase and the Reduce phase. The map phase converts the input data into another set of data (key–value pairs). This new intermediate dataset then serves as the input to the reduce phase. The reduce phase acts on the datasets to combine (aggregate and consolidate) and reduce them to a smaller set of tuples. The result is then stored back in HDFS.
2. **Spark:** It is both a programming model as well as a computing model. It is an open-source big data processing framework. It was originally developed in 2009 at UC Berkeley's AmpLab and became an open-source project in 2010. It is written in Scala. It provides in-memory computing for Hadoop. In Spark, workloads execute in memory rather than on disk owing to which it is much faster (10 to 100 times) than when the workload is executed on disk. However, if the datasets are too large to fit into the available system memory, it can perform conventional disk-based processing. It serves as a potentially faster and more flexible alternative to MapReduce. It accesses data from HDFS (Spark does not have its own distributed file system) but bypasses the MapReduce processing.

   Spark can be used with Hadoop coexisting smoothly with MapReduce (sitting on top of Hadoop YARN) or used independently of Hadoop (standalone). As a programming model, it works well with Scala, Python (it has API connectors for using it with Java or Python) or R programming language.

   The following are the Spark libraries:

   a) ***Spark SQL:*** Spark also has support for SQL. Spark SQL uses SQL to help query data stored in disparate applications.
   b) ***Spark streaming:*** It helps to analyze and present data in real time.
   c) ***MLib:*** It supports machine learning such as applying advanced statistical operations on data in Spark Cluster.
   d) ***GraphX:*** It helps in graph parallel computation.

   Spark and Hadoop are usually used together by several companies. Hadoop was primarily designed to house unstructured data and run batch processing operations on it. Spark is used extensively for its

high speed in memory computing and ability to run advanced real-time analytics. The two together have been giving very good results.

Hadoop Ecosystem Components for Data Analysis

1. **Pig:** It is a high-level scripting language used with Hadoop. It serves as an alternative to MapReduce. It has two parts:
   (a) ***Pig Latin:*** It is SQL-like scripting language. Pig Latin scripts are translated into MapReduce jobs which can then run on YARN and process data in the HDFS cluster. It was initially developed by Yahoo. It is immensely popular with developers who are not comfortable with MapReduce. However, SQL developers may have a preference for Hive.

   How it works? There is a "Load" command available to load the data from "HDFS" into Pig. Then one can perform functions such as grouping, filtering, sorting, joining etc. The processed or computed data can then be either displayed on screen or placed back into HDFS.

   It gives you a platform for building data flow for ETL (Extract, Transform and Load), processing and analyzing huge data sets.

   (b) ***Pig runtime:*** It is the runtime environment.
2. **Hive:** Hive is a data warehouse software project built on top of Hadoop. Three main tasks performed by Hive are summarization, querying and analysis. It supports queries written in a language called HQL or HiveQL which is a declarative SQL-like language. It converts the SQL-style queries into MapReduce jobs which are then executed on the Hadoop platform.

Difference between Hive and RDBMS

Both Hive and traditional databases such as MySQL, MS SQL Server, PostgreSQL support SQL interface. However, Hive is better known as a datawarehouse (D/W) rather than a database.

Let us look at the difference between Hive and traditional databases as regards the schema.

1. Hive enforces schema on Read Time whereas RDBMS enforces schema on Write Time. In RDBMS, at the time of loading/inserting data, the table's schema is enforced. If the data being loaded does not conform to the schema then it is rejected. Thus, the schema is enforced on write (loading the data into the database). Schema on write takes longer to load the data into the database; however it makes up for it during data retrieval with a good query time performance. However, Hive does not enforce the schema when the data is being loaded into the D/W. It is enforced only when the data is being read/retrieved. This is called schema on read. It definitely makes for fast initial load as the data load or insertion operation is just a file copy or move.
2. Hive is based on the notion of write once and read many times whereas the RDBMS is designed for read and write many times.
3. Hadoop is a batch-oriented system. Hive, therefore, is not suitable for OLTP (Online Transaction Processing) but, although not ideal, seems closer to OLAP (Online Analytical Processing). The reason being that there is quite a latency between issuing a query and receiving a reply as the query written in HiveQL will be converted to MapReduce jobs which are then executed on the Hadoop cluster. RDBMS is suitable for housing day-to-day transaction data and supports all OLTP operations with frequent insertions, modifications (updates), deletions of the data.

4. Hive handles static data analysis which is non-real-time data. Hive is the data warehouse of Hadoop. There are no frequent updates to the data and the query response time is not fast. RDBMS is suited for handling dynamic data which is real time.
5. Hive can be easily scaled at a very low cost when compared to RDMS. Hive uses HDFS to store data, thus it cannot be considered as the owner of the data, while on the other hand RDBMS is the owner of the data responsible for storing, managing and manipulating it in the database.
6. Hive uses the concept of parallel computing, whereas RDBMS uses serial computing.

We summarize the difference in Table 4.5.

Table 4.5 Hive versus RDBMS

	Hadoop	RDBMS
Data Variety	Used for structured, semi-structured and unstructured data. Hadoop supports a variety of data formats in real time such as XML, JSON, and text-based flat file formats.	Used for structured data
Data Storage	Usually datasets of size terabytes, petabytes	Usually datasets of size gigabytes
Querying	HiveQL	SQL
Query Response	In Hadoop, there is latency due to batch processing.	In RDBMS, query response time is immediate.
Schema	Schema required on read	Schema required on write
Speed	Writes are faster compared to reads as there is no adherence to schema required at the time of inserting or writing data. Schema is enforced at read time	Reads are very fast (supported by building indexes on required columns).
	Hadoop is designed for write once read many times. It does not work for random reading and writing of a few records like RDBMS.	RDBMS is designed for read and write many times.
Cost	Apache Hadoop is open-source, large-scale, distributed, scalable, data intensive computing.	Available as proprietary RDBMS such as Oracle, MS SQL Server, IBM DB2, etc. Also open-source RDBMS are available such as MySQL, PostgreSQL, etc.
Use Cases	Analytics, data discovery	OLTP (Online Transaction Processing). Mainly used to store and process day-to-day business data.

(Continued)

Table 4.5 (Continued)

	Hadoop	RDBMS
Throughput	High	Low
Scalability	Horizontal (Hadoop scales by adding nodes to a Hadoop cluster of easily available commodity machines).	Vertical: RDBMS scales vertically by increasing the horsepower (CPU, Hard Disk Capacity, RAM, etc.) of the machine.
Hardware	Commodity/Utility Hardware	High End Servers
Integrity	Low	High. Obeys ACID properties A – Atomicity C – Consistency I – Integrity D – Durability

Difference between Hive and HBase

1. Hive is a MapReduce-based SQL engine that runs on top of Hadoop. HBase is a key–value NoSQL database that runs on top of HDFS.
2. Hive is for batch processing of big data. HBase is for real-time data streaming.

Impala

It is a high performance SQL engine that runs on Hadoop cluster. It is ideal for interactive analysis. It has very low latency measured in milliseconds. It supports a dialect of SQL called Impala SQL.

ZooKeeper

It is a coordination service for distributed applications.

Oozie

It is a workflow scheduler system to manage Apache Hadoop jobs.

Mahout

It is a scalable machine learning and data mining library.

Chukwa

It is a data collection system for managing large distributed systems.

Ambari

It is a web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters.

4.2.5 Hadoop Distributions

Hadoop is an open-source Apache project. Anyone can freely download the core aspects of Hadoop. The core aspects of Hadoop include the following:

1. Hadoop Common
2. **H**adoop **D**istributed **F**ile **S**ystem (HDFS)
3. Hadoop YARN (**Y**et **A**nother **R**esource **N**egotiator)
4. Hadoop MapReduce

There are few companies such as IBM, Amazon Web Services, Microsoft, Teradata, Hortonworks, Cloudera, etc. that have packaged Hadoop into a more easily consumable distributions or services. Although each of these companies have a slightly different strategy, the key essence remains its ability to distribute data and workloads across potentially thousands of servers thus making big data manageable data. A few Hadoop distributions are given in Figure 4.12.

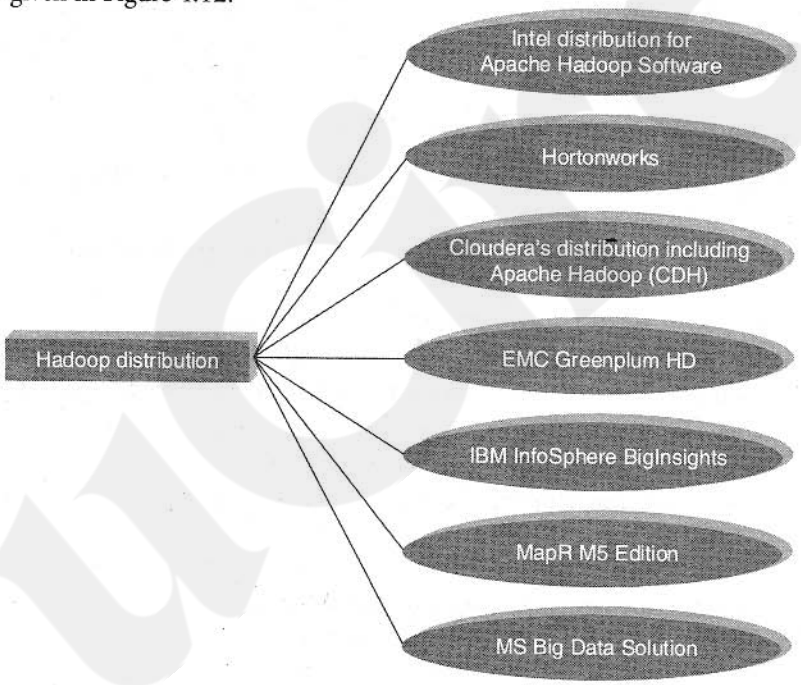


Figure 4.12 Hadoop distributions.

4.2.6 Hadoop versus SQL

Table 4.6 lists the differences between Hadoop and SQL.

Table 4.6 Hadoop versus SQL

Hadoop	SQL
Scale out	Scale up
Key–Value pairs	Relational table
Functional Programming	Declarative Queries
Offline batch processing	Online transaction processing

4.2.7 Integrated Hadoop Systems Offered by Leading Market Vendors

Refer Figure 4.13 to get a glimpse of the leading market vendors offering integrated Hadoop systems.

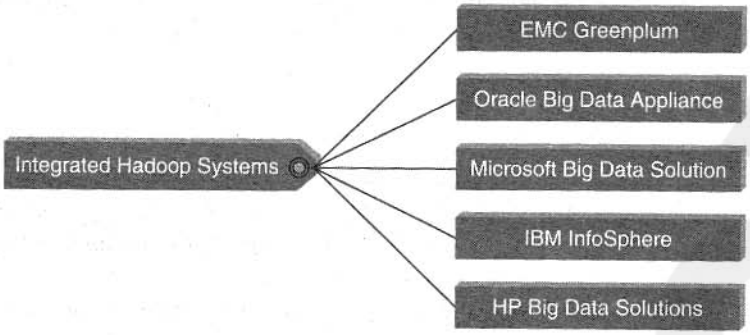


Figure 4.13 Integrated Hadoop systems.

4.2.8 Cloud-Based Hadoop Solutions

Amazon Web Services holds out a comprehensive, end-to-end portfolio of cloud computing services to help manage big data. The aim is to achieve this and more along with retaining the emphasis on reducing costs, scaling to meet demand, and accelerating the speed of innovation.

The Google Cloud Storage connector for Hadoop empowers one to perform MapReduce jobs directly on data in Google Cloud Storage, without the need to copy it to local disk and running it in the Hadoop Distributed File System (HDFS). The connector simplifies Hadoop deployment, and at the same time reduces cost and provides performance comparable to HDFS, all this while increasing reliability by eliminating the single point of failure of the name node. Refer Figure 4.14.

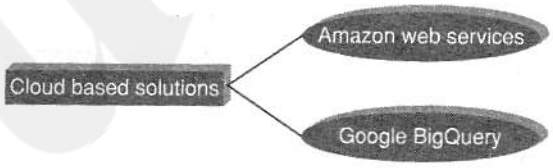


Figure 4.14 Cloud-based solutions.