**Data Curation**

**3.1 Introduction**

**3.1.1 What is Data Curation?**

Data Curation is the way through which data management has been done in order to engage user in data discovery and analysis.

Data Curation is the active and ongoing management of data.

Data curation activities enable data discovery and retrieval of data , maintain quality of data, add valuable insights, and provide data for re-use over time.

Data Curation is an iterative process which includes three main stages:

* Preserving : Collection of data and taking care of it. i.e. manage the data and store it.
* Sharing : To make data available across various domain according to the need and use potential of data as the domain required.
* Discovering : It deals with reusing of existing data with different combinations and generating new data.

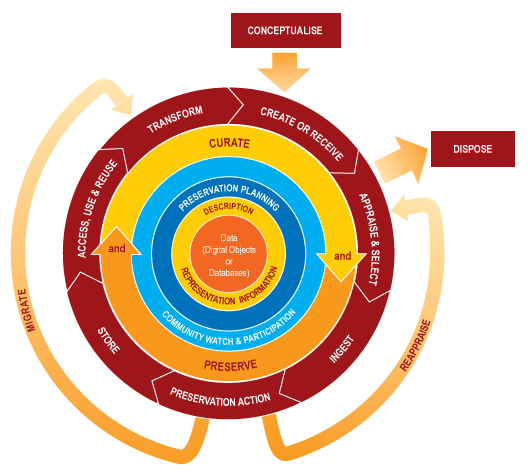
Data curation is more concerned with maintaining and managing the metadata rather than the database itself.

The process of data curation revolves around ingesting metadata such as schema, table and column popularity, usage popularity, top joins/filters/queries.

Data curators not only create, manage, and maintain data, but may also be involved in determining best practices for working with that data.

Data curators often present the data in a visual format such as a chart, dashboard or report.

**3.1.2 Data Curation Life Cycle :**



**ACTIONS OF THE DATA CURATION LIFECYCLE**

**1.Describe and Represent Information**

In order to describe metadata use appropriate standards so that it can be controlled over the long term

Proper format can be used for the better representation of all the metadata and associate digital material.

**2.Build Preservation Strategy**

It is important to plan for preservation throughout the data lifecycle

Planning for preservation requires the careful management and administration of data creation during its lifecycle

**3.Collaborate, Supervise, and Participate**

Supervise data creation activities and assist in the creation of the standards to be used, the tools to create data and appropriate software to create it

This action is an excellent place for librarians or archivists, as they can assist in the collaborative and managerial duties of ensuring that data is created appropriately and preserved

**Curate and Preserve**

Take managerial and administrative actions that will promote curation and preservation throughout the lifecycle. Monitoring the datacreation and encouraging best practices through policies and standards will improve the organization of data throughout its life cycle

**Plan your Data Creation**

Build a strategy and policy that will address how data will be captured and stored

**Create OR Receive**

Create data using descriptive and technical metadata; also include preservation metadata if appropriate

Build a collecting policy in order to be prepared to receive data from data creators, other archives and data centers

**Appraise and Select**

Create an appraisal and selection policy with data creators and curators. Once this policy has been established, evaluate the data and select for long-term curation and preservation

**Take in and Transfer**

After appraisal and selection have been completed, transfer the data to an archive or repository and adhere to the guidelines that were created to ensure the activity is completed properly

**Preservation Measures**

Take steps to ensure the long-term preservation of authoritative nature of data

From an archivist’s perspective, data must remain authentic, reliable and usable while maintaining its integrity.

**Store for Access, Reuse and Retrieval**

Store data using appropriate standards to make sure they remain usable and can be retrieved easily

**Transform**

A very important component, this requires creating new data from the original material. Transforming data can mean turning the material into a different format or creating a subset of results

**3.2 Query languages and Operations to specify and transform data**

This topic gives an overview about query language and operations of query languages applied on data in order to transform data.

**3.2.1 Query Language**

Query language is any computer language that send queries to requests and retrieves data from database and information system. For example, SQL(Structured Query Language), TMQL(Topic Map Query Language), XQuery, Xpath, GraphQL etc.

Query Language is a technique of accessing data from the database. Its mainly of two types :

1. **Procedural Query Language**

It gives formal way to access data from database. This gives information about what data is to be accessed and how it has to be accessed from the database along with queries. Relational Algebra is an example of Procedural Query Language.

1. **Non Procedural Query Language**

It gives informal way to access data from database. This gives information only about what data is to be accessed from the database but how it has to be retrieved is unknown. Structured Query Language (SQL) and Query ByExample (QBE) are examples of Non-procedural query language.

**3.2.1 Relational Algebra (Procedural Query Language)**

Relational algebra is a procedural Query language, which takes input as relation and gives relation as a output. Relational algebra uses operator to perform queries, which are of two types unary and binary.

The operations performed by Relational Algebra are as follows:

* Select
* Project
* Union
* Set Difference
* Cartesian Product
* Rename

## Select Operation (σ)

It is used to select subset of tuple which satisfies the given condition. Sigma(σ)Symbol denotes Select operation.

**Notation** − σ*p*(r)

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like − =, ≠, ≥, < ,  >,  ≤.

**Example 1**

σsubect=”Database”(Books)

Output - Selects tuples from Books where subject = 'Database'.

## Projection(π)

It is used to project column(s) satisfy the given condition.Pi(π)Symbol denotes Projection operation.

Notation − ∏A1, A2, An (r)

Where A1, A2 , An are attribute names of relation **r**.

Duplicate rows are automatically eliminated, as relation is a set.

**For example** −

∏subject, author (Books)

Output - Selects and projects columns named as subject and author from the relation Books.

## Union Operation (∪)

It is used to performs binary union between two given relations and is defined as – r ∪ s = { t | t ∈ r or t ∈ s}

**Notation** − r U s

Where **r** and **s** are either database relations or relation

For a union operation to be valid, the following conditions must hold −

* **r**, and **s** must have the same number of attributes.
* Attribute domains must be compatible.
* Duplicate tuples are automatically eliminated.

**For example** −

∏ author (Books) ∪∏author (Articles)

Output − Projects the names of the authors who have either written a book or an article or both.

## Set Difference (−)

The output of the set difference relation are the tuples which are present in one relation but are not in the second relation.

**Notation** − **r** − **s**

Finds all the tuples that are present in **r** but not in **s**.

**For example -**

∏ author (Books) − ∏ author (Articles)

Output –reterives the name of authors who have written books but not articles.

## Cartesian Product (Χ)

It is used to combine information from two different relations or tables into one single relation.

**Notation** − r Χ s

Where **r** and **s** are relations and their output will be defined as −

r Χ s = { q t | q ∈ r and t ∈ s}

**For example -**

σauthor = 'Balagurusamy'(Books Χ Articles)

Output – Gives a relation, which shows all the books and articles written by author

‘Balagurusamy’.

## Rename Operation (ρ)

The output of relational algebra queries produce relation without name. so this operation, is used to rename produced output relation.

**Notation** − *ρ* x (E)

Where x is the name given to the result produced by expression E.

**3.2.2 Joins**

Join is used to combine Cartesian product followed by a selection process. If the given condition is satisfied, a join operation pairs two tuples from different relations.

Joins are of following types :

## Theta (θ) Join

Theta join combines tuples from two different relations if the given theta condition is satisfied. The join condition is denoted by the symbol **θ**.

### Notation -

R1 ⋈θ R2

R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. ,Bn) such that the attributes don’t have anything in common, that is R1 ∩ R2 = Φ.

**Note** - Theta join can use all kinds of comparison operators.

## Natural Join (⋈)

Like Theta(θ) join, Natural join does not use any comparison operator.

It does not concatenate the way a Cartesian product does. We can perform a Natural Join only if there is at least one common attribute that exists between two relations.

Theta(θ) join and Natural join are also called as inner joins.

An inner join includes only those tuples with matching attributes and the rest are discarded in the resulting relation.

1. **Outer Join**

To overcome unmatched attribute omission of inner join, we have outer joins to include all the tuples from the participating relations in the resulting relation.

Three kinds of outer joins:

* Left outer join
* Right outer join
* Full outer join.

## Left outer join(R ⋈S)

All the tuples from the Left relation, R, are included in the output relation. If there are tuples in R which does not have matching tuple in the Right relation S, then the S-attributes in the output relation are made NULL.

## Right outer join (R ⋈S)

All the tuples from the Right relation, S, are included in the output relation. If there are tuples in S which does not have matching tuple in the Left relation R, then the R-attributes in the output relation are made NULL.

## Full outer join (R ⋈S)

All the tuples from Left and Right relations are included in the output relation. If there are no matching tuples for both relations, then their respective unmatched attributes are made NULL.

**3.2.3Aggregate / Group Functions**

In database management an aggregate function is a function in which the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Aggregate Functions are:

* Count()
* Sum()
* Avg()
* Min()
* Max()

1. Count()

**Count(\*):** It returns total number of records present in a given relation.  
**Count(cloumnname):** It returns number of Non Null values over the given

columnname.

For example, **Count(salary):** Return number of Non Null values over the column salary.

1. Sum()

This function returns the sum of non-null values over thegiven attribute.

For example, **sum(salary):** Returns Sum of all Non Null values of Column salary.

1. Avg()

This function returns the average of the values over the given attribute.

For example, **Avg(salary):**Returns Avg of all values of column salary.

1. Min()

This function returns minimum value of a given attribute.

For example, **Min(salary):**Returnsminimum value in the salary column except NULL.

1. Max()

This function returns maximum value of a given attribute.

For example, ,**Max(salary):****):**Returnsmaximum value in the salary column.

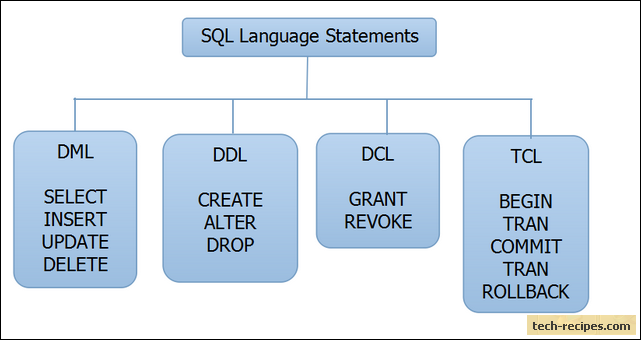
**3.2.1 Structured Query Language (SQL, Non Procedural Query Language)**

SQL(Structured Query Language) is a non-procedural Query Language, and used to have communication with a database. It is a standard language used for storing, manipulating and retrieving data in database.

The four main categories of SQL statements are as follows:

**1.**DDL (Data Definition Language) **2**.DML (Data Manipulation Language)

**3.**DCL (Data Control Language) **4.**TCL (Transaction Control Language)



1. **DDL (Data Definition Language)**

DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

**CREATE** – It is used to create a new Table, database, schema  
**ALTER** – It is used to alter existing table, column description  
**DROP** – It is used to delete existing objects from database

1. **DML (Data Manipulation Language)**

DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

DML statements include the following:

**SELECT** – It is used to select records from a table  
**INSERT** – It is used to insert new records  
**UPDATE** – It is used to update/Modify existing records  
**DELETE** – It is used to delete existing records

1. **DCL (Data Control Language)**

DCL statements control the level of access that users have on database objects.

**GRANT** – It is used to allows users to read/write on certain database objects  
**REVOKE** –It is used to keeps users from read/write permission on database objects

1. **TCL (Transaction Control Language)**

TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.

**BEGIN Transaction** – It is used to opens a transaction  
**COMMIT Transaction** – It is used to commits a transaction  
**ROLLBACK Transaction** – It is used to ROLLBACK a transaction in case of any error

**3.3 Structured/schema based systems as users and acquirers of data**

Structured data is a data used by structured system. Structured data is defined by a data model, has a well define structure and it is easily used and accessed by the users and a computer program.

Structured data is usually stored in well-defined schemas such as Databases. It is stored in tabular format with column and rows that clearly define its attributes.

SQL (Structured Query language) is often used to manage and access structured data stored in databases.

**Characteristics of Structured Data:**

* Data conforms to a data model and has easily identifiable structure
* Data is stored in the form of rows and columns  
  Example : Database
* Data is well organized so, Definition, Format and Meaning of data is explicitly known
* Data resides in fixed fields within a record or file
* Similar entities are grouped together to form relations or classes
* Entities in the same group have same attributes
* Easy to access and query, So data can be easily used by other programs
* Data elements are addressable, so efficient to analyse and process

**Sources of Structured Data:**

* **SQL Databases** :The SQL database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

**For example:** The college Database organizes the data about the admin, staff, students and faculty etc.

* **Spreadsheets such as Excel**

*Spreadsheet are Excel reports* display highly formatted, summarized data and are often designed as presentation tools for management or executive users. A typical spreadsheet report makes judicious use of empty space for formatting, repeats data for aesthetic purposes, and presents only high-level analysis.

Datasets typically used in Excel come in three fundamental forms:

* The spreadsheet report
* The flat data file
* The tabular dataset
* **OLTP Systems**

*OLTP* (Online Transactional Processing) is a category of data processing that is focused on transaction-oriented tasks. OLTP typically involves inserting, updating, and/or deleting small amounts of data in a database.

OLTP mainly deals with large numbers of transactions by a large number of users.

## Examples of OLTP Transactions includes:

* 1. Online banking
  2. Purchasing a book online
  3. Booking an airline ticket
  4. Sending a text message
  5. Order entry
* **Online forms**

Google Search works hard to understand the content of a page. Structured data is used to provide information about a page and page content has been classified.

Google uses structured data that to understand the content of the page, as well as to gather information about the web and the world in general.

* **Sensors such as GPS or RFID tags**

The GPS (**Global Positioning System**) gives location accuracy.

**RFID** is an acronym for “radio-frequency identification” and refers to a technology whereby digital data encoded in **RFID**tags or smart labels (defined below) are captured by a reader via radio waves.

The above two are the integration of emerging wireless remote **sensor data which generate structured data.**

* **Medical devices**

Medical devices are also generate structured data.

**e.g**patient-monitoring medical devices

**Note - Examples** of machine **generated data** include the following: **Data**from sensors such as GPSs, RFID tags, **medical devices**, **data** from network and web logs, retail and ecommerce **data**.

**Advantages of Structured Data:**

* Due to well-defined structure, Structured data helps in easy storage and access of data
* Data can be indexed based on text string as well as attributes. This makes search operation hash-free
* Data mining is easy i.e knowledge can be easily extracted from data
* Due to well structured form of data operations such as Updating and deleting is easy.
* Business Intelligence operations such as Data warehousing can be easily undertaken
* Structured data is easily scalable in case there is an increment of data

**Note:**Structured data accounts for only about 20% of data but because of its high degree of organization and performance make it foundation of Big data.

**3.4Semi-structuredsystems as users and acquirers of data**

[Semi-structured data](https://www.datamation.com/big-data/semi-structured-data.html) has its own internal tags and markings that identify it as a separate data elements, which allows information grouping and hierarchies. Both documents and databases can be semi-structured. Though semi-structured data represents about 5-10% of the structured/semi-structured/unstructured data pie, this data has critical business usage cases.

A very common example of a semi-structured data type is Email.

Examples of Semi-structured Data

* **Markup language XML**

This is a semi-structured document language. XML is a set of document encoding rules that defines a human- and machine-readable format.

Xml allows user to define their own tags.It is main source of data for business usage.

* **Open standard JSON (JavaScript Object Notation)**

JSON is another semi-structured data interchange format.JSONis used to transmit data between web applications and servers.

* **NoSQL**

NoSQL (“not only SQL”) databases contain data in a Semi-structured format. NoSQL databases differ from relational databases because they do not separate the organization (schema) from the data. This makes NoSQL a better choice to store information that does not easily fit into the record and table format, such as text with varying lengths. It also allows for easier data exchange between databases. Some newer NoSQL databases like [MongoDB](https://www.mongodb.com/) and [Couchbase](https://www.couchbase.com/) also incorporate semi-structured documents by natively storing them in the JSON format.

**3.5 Unstructured systems in the acquisition and structuring of data**

Unstructured data is not structured because it does not have pre-defined data models or schema.Unstructured data has its internal. It may be textual or non-textual, and human- or machine-generated. It may also be stored within a non-relational database like NoSQL.

Typical human-generated unstructured data includes:

* **Text files:** Word processing, spreadsheets, presentations, email, logs.
* **Email:** Email has some internal structure thanks to its metadata, and we sometimes refer to it as [semi-structured](https://www.datamation.com/big-data/semi-structured-data.html). However, its message field is unstructured and traditional [analytics tools](https://www.datamation.com/big-data/top-12-big-data-tools.html) cannot parse it.
* **Social Media:** Data from Facebook, Twitter, LinkedIn.
* **Website:** YouTube, Instagram, photo sharing sites.
* **Mobile data:** Text messages, locations.
* **Communications:** Chat, IM, phone recordings, collaboration software.
* **Media:** MP3, digital photos, audio and video files.
* **Business applications:** MS Office documents, productivity applications.

Typical machine-generated unstructured data includes:

* **Satellite imagery:** Weather data, land forms, military movements.
* **Scientific data:** Oil and gas exploration, space exploration, seismic imagery, atmospheric data.
* **Digital surveillance:** Surveillance photos and video.
* **Sensor data:** Traffic, weather, oceanographic sensors.

**3.6 Security and ethical considerations in relation to authenticating and authorizing access to data on remote system**

Security is one of the most important parameter for data system.

Data security is used to protect digital data, applied to prevent unauthorized access to computers, database and websites.

Data security also protects data from corruption.

Data security is also known as information security (IS) or computer security.

In today’s data oriented system, we face various types of security attacks

1. [**Denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks**](https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#Denial-of-service%20(DoS)%20and%20distributed%20denial-of-service%20(DDoS)%20attacks)

A denial-of-service attack overwhelms a system’s resources so that it cannot respond to service requests.

1. [**Man-in-the-middle (MitM) attack**](https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#Man-in-the-middle%20(MitM)%20attack)

A MitM attack occurs when a hacker inserts itself between the communications of a client and a server. It causes misuse of data.

1. [**Phishing and spear phishing attacks**](https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#Phishing%20and%20spear%20phishing%20attacks)

Phishing attack is the act of sending emails that appear to be from trusted sources in-order to get personal information or influencing users to do something.

1. [**Drive-by attack**](https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#Drive-by%20attack)

Drive-by download attacks is used to spread malware. Hackers look for insecure websites and plant a malicious script into HTTP or PHP code on one of the pages. This script might install malware directly onto the computer of someone who visits the site, or it might re-direct the victim to a site controlled by the hackers. Drive-by downloads can happen when visiting a website or viewing an email message or a pop-up window.

1. [**Password attack**](https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#Password%20attack)

Because passwords are used to authenticate users to an information system, obtaining passwords is a common and effective attack approach. Access to a person’s password can be obtained by looking around the person’s desk, ‘‘sniffing’’ the connection to the network to acquire unencrypted passwords, using social engineering, gaining access to a password database or outright guessing.

1. [**SQL injection attack**](https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#SQL%20injection%20attack)

SQL injection has become a common issue with database-driven websites. It occurs when a malefactor executes a SQL query to the database via the input data from the client to server.

1. [**Cross-site scripting (XSS) attack**](https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#Cross-site%20scripting%20(XSS)%20attack)

XSS attacks use third-party web resources to run scripts in the victim’s web browser or scriptable application.

1. [**Malware attack**](https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#Malware%20attack)

Malicious software can be described as unwanted software that is installed in your system without your consent.

Examples of data security technologies include data backups, data masking and data erasure. A key data security technology measure is encryption, where digital data, software/hardware, and hard drives are encrypted so that it is made unreadable to unauthorized users and hackers.

One of the most commonly used methods for data security is the use of authentication and authorization.

With authentication, users must provide a password, code, biometric data, or some other form of data to verify identity of user before we grant access to a system or data.

There are various major process used for security of data on remote system.

1. **Authentication**

Authentication is the process of determining whether someone or something is, in fact, who or what it declares itself to be. Authentication technology provides access control for systems by checking to see if a user's credentials match the credentials in a database of authorized users or in a data authentication server.

Users are usually identified with a user ID, and authentication is accomplished by providing user with password.

If a password, that matches with that user ID for a specific user, then the user is allowed to access and grant resources.

Authentication factors can vary from one of the following:

* **Single- Factor Authentication:**This is the simplest form of authentication method.

In this method password is verified against username, if both are verified successfully then user is allowed to access or grant resources, particular system, website or a network.

* **Two- Factor Authentication:**This method of authentication requires a two- step verification process.

This method requires a username and password along with a piece of information only the user knows.

Using a username and password along with a confidential information makes it that much harder for hackers to steal valuable and personal data.

* **Multi- Factor Authentication:**This is the most advanced method of authentication which requires two or more levels of security from independent categories of authentication to grant user access to the system. This type of authentication utilizes factors that are independent of each other in order to eliminate any data exposure as well very hard to hackers to hack the confidential data.

It is commonly used in financial organizations, banks, and law enforcement agencies to use multiple- factor authentication.

1. **Authorization**

Authorization is the process of giving someone permission to do or have something. In multi-user computer systems, a system administrator defines for the system which users are allowed [access](https://whatis.techtarget.com/definition/access) to the system and what privileges of use (such as access to which file directories, hours of access, amount of allocated storage space, and so forth).

Assuming that someone has logged in to a computer [operating system](https://whatis.techtarget.com/definition/operating-system-OS) or [application](https://searchsoftwarequality.techtarget.com/definition/application), the system or application may want to identify what resources the user can be given during this session. Thus, authorization is sometimes seen as both the preliminary setting up of permissions by a system administrator and the actual checking of the permission values that have been set up when a user is getting access.

Logically, authorization is preceded by [authentication](https://searchsecurity.techtarget.com/definition/authentication).

1. **Encrypt your data.**

Data encryption is a security method where information is encoded and can only be accessed or decrypted by a user with the correct encryption key. Encrypted data, also known as ciphertext, appears scrambled or unreadable to a person or entity accessing without permission.

1. **Backup your data.**

In information technology, **a backup**, or **data backup** is **a** copy of computer **data** so that it may be used to restore **the** original after **a data** loss event. ... **A backup**system contains at least one copy of all **data** considered worth saving, **meaning** that**the data** storage requirements can be significant.

1. **Turn off your computer.**

This protects **your computer** by stopping threats from entering **the** system and spreading between devices. It can also help prevent **your** data leaving **your computer**. If **your computer** ports are open, anything coming into them could be processed. This is bad if it's**a** malicious program sent by **a** hacker.

1. **Use firewall**

A **firewall** is a network **security** system designed to prevent unauthorized access to or from a private network. ... Network firewalls are frequently used to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets.

Algorithms used to secure data are as follows:

1. RSA Algorithm
2. DES Algorithm( Data Encryption Standard)
3. AES( [Advanced](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard) Encryption Standard)
4. Public Key Algorithm
5. OTP password Algorithm
6. Data Encryption standard Algorithm

**3.7 Software Development Tools**

Software or application development tools are the set of platform, environment and softwares are needed in order to develop an application.

Usually an application needs more than one tool and a team to develop it, many times the changes has to be done in the previously developed and deployed application or system in such cases software tools for version control or environment like GitHub is required.

Following section explains commonly used version and source control software.

**3.7.1 Version control/Source control**

Source control is part of a larger strategy to take one-off, hodge-podge code and make it well documented and ready for larger scale development. With source control, keeping track of your code doesn’t curtail development. Source control is also called as version control or revision control.

Revision control system maintain the set of all versions of files which are organized in a particular manner along with the time. Furthermore, you allow yourself the freedom to experiment, remove or make drastic changes to your code base knowing full well that you can rewind in time and compare what was changed or even revert back if need be.

You can add the following files to visual studio source control:

* Solution files(\*.sln)
* Project files, for example, \*.csproj, \*.vbproj files.
* Application configuration files, based on XML, used to control run-time behavior of a Visual Studio project.

Files that you cannot add to source control include the following:

* Solution user option files (\*.suo)
* Project user option files, for example, \*.csproj.user, \*.vbproj.user files.
* Web information files, for example, \*.csproj.webinfo, \*.vbproj.webinfo, that control the virtual root location of a web project.
* Build output files, for example, \*.dll and \*.exe files.

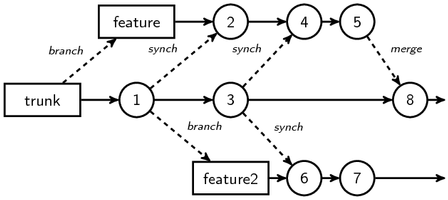
**3.7.1.1 Version Control Terminology/ Functionalities**

The visual studio documentation, uses a number of terms to describe source control features and concepts.

Common terms used in the version control are as follows:

* **Basis version** : The server version of a file from which a local version is derived.
* **Binding** : Information that correlates a working folder for a solution or project on disk to its folder in the database.
* **Branching** : Process of creating a new version, or branch of a shared file or project under source control. Once a branch has been created, the two versions under source control will have a shared history up to a certain point and divergent histories after that point.
* **Conflict** : Two or more different changes to the same line of code in situations where two or more developers have checked out and edited the same file.
* **Connection** : A live data link between a source control client (for example, Visual Studio) and a source control database server.
* **Database** : Location where all master copies, history, project structures, and user information are stored. A project is always contained within one database. Multiple projects can be stored in one database, and multiple databases can be used. Other terms commonly used for a database are repository and store.
* **History** : Record of changes to a file since it was initially added to source control. With version control, you can return to any point in the file history and recover the file as it existed at that point.
* **Label/Tag** : User defined name attached to a specific version of a source-controlled item.
* **Local copy** : File in a user’s working folder to which changes are saved until a check-in occurs. A local copy is sometimes referred to as a working copy.
* **Master copy** : The most recently checked-in version of a source-controlled file, as opposed to the local copy of a file in your working folder. Other terms for master copy are server version and database version.
* **Merging** : Process of combining differences in two or more modified versions of a file into a new file version. Merging can affect different versions of the same file or changes made to the same file version.
* **Shared file** : A file having versions that reside in more than one source control location. Other terms for a shared file are copy and shortcut.
* **Solution root** : An empty folder in a database that contains all items in a source-controlled solution. By default, this folder is <solutionname>.root.
* **Super-unified root** : A virtual container beneath which all projects and files in a source-controlled solution are located. For example [SUR]:\ is the super-unified root of a source-controlled solution containing projects that are located in [SUR]:\C:\Solution\ProjOne and [SUR]:\D:\ProjTwo
* **Unified root** : A path to the parent directory for all working folders and files in a source-controlled solution or project. For example, C:\Solution is the unified root of a source-controlled solution containing files that are located in C:\Solution, C:\Solution\ProjOne and C:\Solution\ProjTwo.
* **Working folder** : Location where your local copies of source-controlled item are stored, usually on your own computer. Another term for a working folder is workspace.
* **Baseline** : It is a version of a document on which we can make the changes. This document is also called as source file.
* **Branch** : It is a point from where the source file is branched and after this two copies of this file will be kept. The branches are also called as forks.
* **Checkout** : It is a process of creating a local working copy form the repository. At any instances of time a user can obtain the working local copy by using the checkout.
* **Commit** : It is an opposite process of checkout and hence can be also called as check-in. In this process the changes which are made in the working local directory are saved back into the repository.
* **Trunk** : It is called as a mainline or the unique line of the development which is not actually a branch.

The overall diagram of source control or version control can be shown in following figure.



**An example of version control flow**

**3.7.2 GitHub**

Github is a platform for code hosting and collaborative development for systems. GitHub removes the location barrier; any member of the team can work on a particular project from any location and can upload or download the files or folders any times.

Recently GitHub is used by many people for sharing the codes and the files.

GitHub is used for doing version control, as simultaneously number of people can upload their part of code and then merge all the sub parts of the code.

GitHub has a central repository where the data for a particular project is stored and using branches many people can link to that project and update their part as and when done.

Git is as version control tool used in GitHub to perform various jobs like fetching data pushing and pulling request to and from the central repository to the user and vice versa.

Following is the procedure followed in GitHub

* Repositories
* Branches
* Commits
* Pull Requests
* Git (the version control software GitHub is built on)

1. **Create a repositories**

A GitHub **repository** can be used to store a development **project**.

It can contain **folders** and any type of **files** (HTML, CSS, JavaScript, Documents, Data, Images).

A GitHub repository should also include a **licence** file and a **README** file about the project.

A GitHub repository can also be used to store ideas, or any resources that you want to share.

1. **Create branches**

A GitHub branch is used to work with different **versions** of a repository at the same time.

By default a repository has a **master** branch (a production branch).

Users those who are going to participate in the development of application are two be connected to the main or central repository branches acts as the connectivity link between central repository and the user.

1. **Make and commit changes**

Once the changes are done they have to updated on the main repository. Commit is used to make that changes and update the central repository.

1. **Open a pull request**

If any changes have to be reflected on particular user’s module then he can make a pull request which is used to get the update from various other members of the team.

1. **Merges your pull request**

Once the development part of every branch is done completely or partially it is merged and stages of the projects are developed.

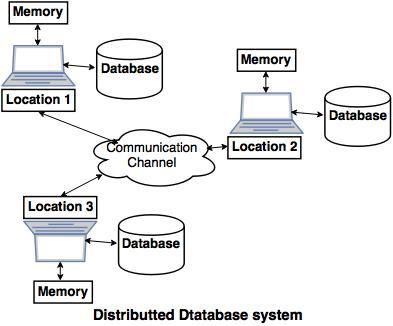
**3.8 Large scale data system**

In order to store large data we cannot use normal database systems like RDBMS and hence databases like NoSQL, MognoDB and HBase etc. are used to store vast amount of data.

Large scale system do not always have centralized data storage system. Large scale data system mostly use distributed data storage.

**3.8.1 Distributed database system**

A distributed data system is not limited to one system, it is spread over different systems i.e. multiple computers having vast data and are interlinked with each other through network.

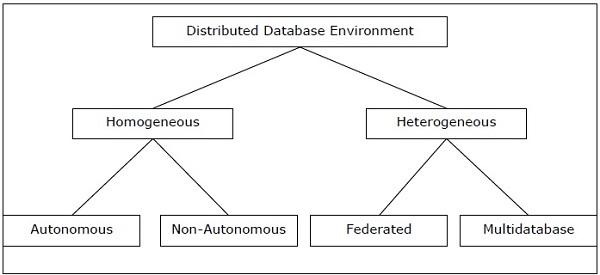


The above diagram is a typical example of distributed database system, in which communication channel is used to communicate with the different locations and every system has its own memory and database.

## Goals of Distributed Database system.

**The concept of distributed database was built with a goal to improve:**  
  
**Reliability:** In distributed database system, if one system fails down or stops working for some time another system can complete the task.  
**Availability:** In distributed database system reliability can be achieved even if sever fails down. Another system is available to serve the client request.  
**Performance:** Performance can be achieved by distributing database over different locations. So the databases are available to every location which is easy to maintain.

**Types of distributed data system**



1. **Homogeneous distributed database**

Homogeneous distributed database system is a network of two or more databases (With same type of DBMS software) which can be stored on one or more machines.

So, in this system data can be accessed and modified simultaneously on several databases in the network. Homogeneous distributed system are easy to handle.

### Types of Homogeneous Distributed Database

There are two types of homogeneous distributed database −

* **Autonomous** − Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.
* **Non-autonomous** − Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

1. **Heterogeneous distributed database**

Heterogeneous distributed database system is a network of two or more databases with different types of DBMS software, which can be stored on one or more machines.

In this system data can be accessible to several databases in the network with the help of generic connectivity (ODBC and JDBC).

### Types of Heterogeneous Distributed Databases

* **Federated** − The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.
* **Un-federated(multi-database)** − The database systems employ a central coordinating module through which the databases are accessed.

**3.8.2 NoSQL**

A NoSQL database is also referred as “non SQL ” or “non relational” database.

It is an open source [distributed, non relational databases](https://en.wikipedia.org/wiki/Distributed_database).

It provides a mechanism for storage and retrieval of data which are not based on RDBMS principle.

NoSQL databases are structured in a key-value pair, graph database, document-oriented or column-oriented structure.

Using NoSQL you can directly start inserting this new data in your existing structure without creating any new pre-defined columns or pre-defined structure.

### Benefits of NoSQL

**Schema Less:**  
NoSQL databases being schema-less do not define any strict data structure.

**Dynamic and Agile:**  
NoSQL databases have good tendency to grow dynamically with changing requirements. It can handle structured, semi-structured and unstructured data.

**Scales Horizontally:**  
In contrast to SQL databases which scale vertically, NoSQL scales horizontally by adding more servers and using concepts of sharding and replication. This behavior of NoSQL fits with the cloud computing services such as Amazon Web Services (AWS) which allows you to handle virtual servers which can be expanded horizontally on demand.

**Better Performance:**  
All the NoSQL databases claim to deliver better and faster performance as compared to traditional RDBMS implementations.

### Types of NoSQL Databases

**Document Oriented Databases**   
Document oriented databases treat a document as a whole and avoid splitting a document in its constituent name/value pairs. At a collection level, this allows for putting together a diverse set of documents into a single collection.

e.g. MongoDB, HBase, Cassandra, Amazon SimpleDB, Hypertable, etc.

**Graph Based Databases**  
A graph database uses graph structures with nodes, edges, and properties to represent and store data. By definition, a graph database is any storage system that provides index-free adjacency. This means that every element contains a direct pointer to its adjacent element and no index lookups are necessary.

e.g. Neo4j, OrientDB, Facebook Open Graph, FlockDB, etc.

**Column Based Databases**  
The column-oriented storage allows data to be stored effectively. It avoids consuming space when storing nulls by simply not storing a column when a value doesn’t exist for that column. Each unit of data can be thought of as a set of key/value pairs, where the unit itself is identified with the help of a primary identifier, often referred to as the primary key. Bigtable and its clones tend to call this primary key the row-key.

e.g. CouchDB, OrientDB, etc.

**Key Value Databases**  
The key of a key/value pair is a unique value in the set and can be easily looked up to access the data. Key/value pairs are of varied types: some keep the data in memory and some provide the capability to persist the data to disk.

e.g. Membase, Redis, MemcacheDB, etc.

**3.8.3 MognoDB**

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Any relational database has a typical schema design that shows number of tables and the relationship between these tables. While in MongoDB, there is no concept of relationship.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

**Collection**

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

**Document**

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

## Advantages of MongoDB

* **Schema less** − MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.

Structure of a single object is clear.

No complex joins.

Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.

* **Ease of scale-out** − MongoDB is easy to scale.

Uses internal memory for storing the (windowed) working set, enabling faster access of data.

## Use of MongoDB

* **Document Oriented Storage** − Data is stored in the form of JSON style documents.
* Index on any attribute
* Replication and high availability
* Auto-sharding
* Rich queries
* Fast in-place updates
* Professional support by MongoDB

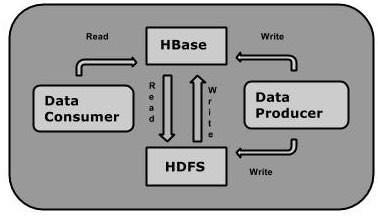
**3.8.4 HBase**

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google’s big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.



**Storage Mechanism in HBase**

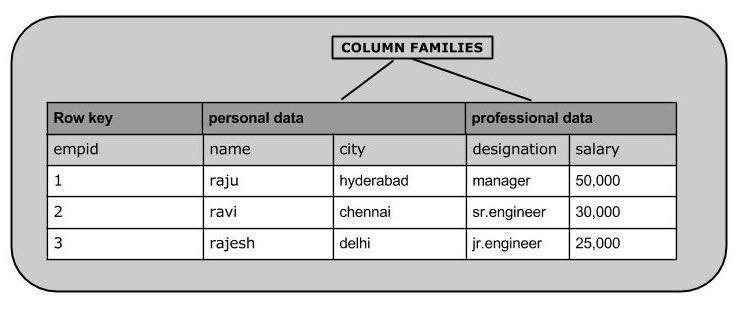
HBase is a **column-oriented database** and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp. In short, in an HBase:

* Table is a collection of rows.
* Row is a collection of column families.
* Column family is a collection of columns.
* Column is a collection of key value pairs.

Given below is an example schema of table in HBase.

| **Rowid** | **Column Family** | | | **Column Family** | | | **Column Family** | | | **Column Family** | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **col1** | **col2** | **col3** | **col1** | **col2** | **col3** | **col1** | **col2** | **col3** | **col1** | **col2** | **col3** |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |  |

The following image shows column families in a column-oriented database:



**Features of HBase**

* HBase is linearly scalable.
* It has automatic failure support.
* It provides consistent read and writes.
* It integrates with Hadoop, both as a source and a destination.
* It has easy java API for client.
* It provides data replication across clusters.

**Applications of HBase**

* It is used whenever there is a need to write heavy applications.
* HBase is used whenever we need to provide fast random access to available data.
* Companies such as Facebook, Twitter, Yahoo, and Adobe use HBase internally.

**3.9 AWS**

**Amazon Web Services (AWS)** is used to provide IT services to the market in the form of web services known as **cloud computing**. These services can instantly spin up hundreds or thousands of servers in minutes and deliver results faster.

AWS is a cost efficient because we pay only for what we use with no long term commitments.

AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that enhance businesses activities.

**What is Cloud Computing?**

**Cloud computing** is an internet-based computing service in which large groups of remote servers are networked to allow centralized data storage, and online access to computer services or resources.

Using cloud computing, organizations can use shared computing and storage resources rather than building, operating, and improving infrastructure on their own.

Cloud computing is a model that enables the following features.

* Users can provision and release resources on-demand.
* Resources can be scaled up or down automatically, depending on the load.
* Resources are accessible over a network with proper security.
* Cloud service providers can enable a pay-as-you-go model, where customers are charged based on the type of resources and per usage.

## Types of Clouds

There are three types of clouds − Public, Private, and Hybrid cloud.

### Public Cloud

In public cloud, the third-party service providers make resources and services available to their customers via Internet. Customer’s data and related security is with the service providers’ owned infrastructure.

### Private Cloud

A private cloud also provides almost similar features as public cloud, but the data and services are managed by the organization or by the third party only for the customer’s organization. In this type of cloud, major control is over the infrastructure so security related issues are minimized.

### Hybrid Cloud

A hybrid cloud is the combination of both private and public cloud. The decision to run on private or public cloud usually depends on various parameters like sensitivity of data and applications, industry certifications and required standards, regulations, etc.

## Cloud Service Models

There are three types of service models in cloud − IaaS, PaaS, and SaaS.

### IaaS

IaaS stands for **Infrastructure as a Service**. It provides users with the capability to provision processing, storage, and network connectivity on demand. Using this service model, the customers can develop their own applications on these resources.

### PaaS

PaaS stands for **Platform as a Service**. Here, the service provider provides various services like databases, queues, workflow engines, e-mails, etc. to their customers. The customer can then use these components for building their own applications. The services, availability of resources and data backup are handled by the service provider that helps the customers to focus more on their application's functionality.

### SaaS

SaaS stands for **Software as a Service**. In this the third-party providers provide end-user applications to their customers with some administrative capability at the application level, such as the ability to create and manage their users. Also some level of customizability is possible such as the customers can use their own corporate logos, colors, etc.

## Advantages of Cloud Computing

Here is a list of some of the most important advantages that Cloud Computing has to offer −

* **Cost-Efficient** − Building our own servers and tools is time-consuming as well as expensive as we need to order, pay for, install, and configure expensive hardware, long before we need it. However, using cloud computing, we only pay for the amount we use and when we use the computing resources. In this manner, cloud computing is cost efficient.
* **Reliability** − A cloud computing platform provides much more managed, reliable and consistent service than an in-house IT infrastructure. It guarantees 24x7 and 365 days of service. If any of the server fails, then hosted applications and services can easily be transited to any of the available servers.
* **Unlimited Storage** − Cloud computing provides almost unlimited storage capacity, i.e., we need not worry about running out of storage space or increasing our current storage space availability. We can access as much or as little as we need.
* **Backup & Recovery** − Storing data in the cloud, backing it up and restoring the same is relatively easier than storing it on a physical device. The cloud service providers also have enough technology to recover our data, so there is the convenience of recovering our data anytime.
* **Easy Access to Information** − Once you register yourself in cloud, you can access your account from anywhere in the world provided there is internet connection at that point. There are various storage and security facilities that vary with the account type chosen.

## Disadvantages of Cloud Computing

Although Cloud Computing provides a wonderful set of advantages, it has some drawbacks as well that often raise questions about its efficiency.

### Security issues

Security is the major issue in cloud computing. Storing data and important files on external service providers always in a risk.

AWS cloud infrastructure is designed to be the most flexible and secured cloud network. It provides scalable and highly reliable platform that enables customers to deploy applications and data quickly and securely.

### Technical issues

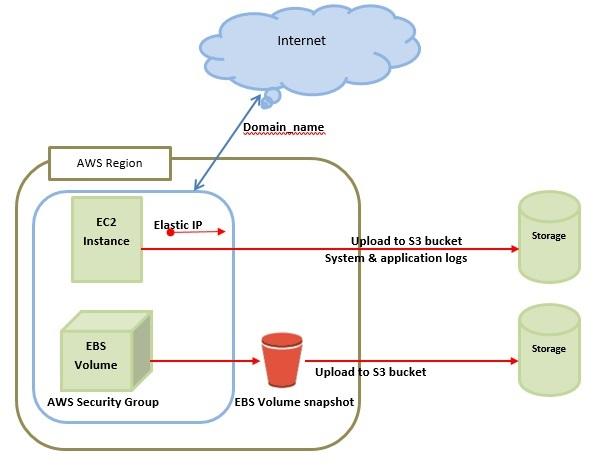
As cloud service providers offer services to number of clients each day, sometimes the system can have some serious issues leading to business processes temporarily being suspended. Additionally, if the internet connection is offline then we will not be able to access any of the applications, server, or data from the cloud.

### Not easy to switch service providers

Cloud service providers promises vendors that the cloud will be flexible to use and integrate, however switching cloud services is not easy. Most organizations may find it difficult to host and integrate current cloud applications on another platform. Interoperability and support issues may arise such as applications developed on Linux platform may not work properly on Microsoft Development Framework (.Net).

# **3.9.2 Amazon Web Services - Basic Architecture**

This is the basic structure of **AWS EC2**, where **EC2** stands for Elastic Compute Cloud. EC2 allow users to use virtual machines of different configurations as per their requirement. It allows various configuration options, mapping of individual server, various pricing options, etc. Following is the diagrammatic representation of the architecture.

****

**Note** − In the above diagram **S3** stands for Simple Storage Service. It allows the users to store and retrieve various types of data using API calls. It doesn’t contain any computing element.

### Load Balancing

**Load balancing** simply means to hardware or software load over web servers, that improver's the efficiency of the server as well as the application. Following is the diagrammatic representation of AWS architecture with load balancing.

Hardware load balancer is a very common network appliance used in traditional web application architectures.

AWS provides the Elastic Load Balancing service, it distributes the traffic to EC2 instances across multiple available sources, and dynamic addition and removal of Amazon EC2 hosts from the load-balancing rotation.

**Elastic Load Balancing** can dynamically grow and shrink the load-balancing capacity to adjust to traffic demands and also support sticky sessions to address more advanced routing needs.

### Amazon Cloud-front

It is responsible for content delivery, i.e. used to deliver website. It may contain dynamic, static, and streaming content using a global network of edge locations. Requests for content at the user's end are automatically routed to the nearest edge location, which improves the performance.

Amazon Cloud-front is optimized to work with other Amazon Web Services, like Amazon S3 and Amazon EC2. It also works fine with any non-AWS origin server and stores the original files in a similar manner.

In Amazon Web Services, there are no contracts or monthly commitments. We pay only for as much or as little content as we deliver through the service.

### Elastic Load Balancer

It is used to spread the traffic to web servers, which improves performance. AWS provides the Elastic Load Balancing service, in which traffic is distributed to EC2 instances over multiple available zones, and dynamic addition and removal of Amazon EC2 hosts from the load-balancing rotation.

Elastic Load Balancing can dynamically grow and shrink the load-balancing capacity as per the traffic conditions.

### Security Management

Amazon’s Elastic Compute Cloud (EC2) provides a feature called security groups, which is similar to an inbound network firewall, in which we have to specify the protocols, ports, and source IP ranges that are allowed to reach your EC2 instances.

Each EC2 instance can be assigned one or more security groups, each of which routes the appropriate traffic to each instance. Security groups can be configured using specific subnets or IP addresses which limits access to EC2 instances.

### Elastic Caches

Amazon Elastic Cache is a web service that manages the memory cache in the cloud. In memory management, cache has a very important role and helps to reduce the load on the services, improves the performance and scalability on the database tier by caching frequently used information.

### Amazon RDS

Amazon RDS (Relational Database Service) provides a similar access as that of MySQL, Oracle, or Microsoft SQL Server database engine. The same queries, applications, and tools can be used with Amazon RDS.

It automatically patches the database software and manages backups as per the user’s instruction. It also supports point-in-time recovery. There are no up-front investments required, and we pay only for the resources we use.

### Hosting RDMS on EC2 Instances

Amazon RDS allows users to install RDBMS (Relational Database Management System) of your choice like MySQL, Oracle, SQL Server, DB2, etc. on an EC2 instance and can manage as required.

Amazon EC2 uses Amazon EBS (Elastic Block Storage) similar to network-attached storage. All data and logs running on EC2 instances should be placed on Amazon EBS volumes, which will be available even if the database host fails.

Amazon EBS volumes automatically provide redundancy within the availability zone, which increases the availability of simple disks. Further if the volume is not sufficient for our databases needs, volume can be added to increase the performance for our database.

Using Amazon RDS, the service provider manages the storage and we only focus on managing the data.

### Storage & Backups

AWS cloud provides various options for storing, accessing, and backing up web application data and assets. The Amazon S3 (Simple Storage Service) provides a simple web-services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web.

Amazon S3 stores data as objects within resources called **buckets**. The user can store as many objects as per requirement within the bucket, and can read, write and delete objects from the bucket.

Amazon EBS is effective for data that needs to be accessed as block storage and requires persistence beyond the life of the running instance, such as database partitions and application logs.

Amazon EBS volumes can be maximized up to 1 TB, and these volumes can be striped for larger volumes and increased performance. Provisioned IOPS volumes are designed to meet the needs of database workloads that are sensitive to storage performance and consistency.

Amazon EBS currently supports up to 1,000 IOPS per volume. We can stripe multiple volumes together to deliver thousands of IOPS per instance to an application.

### Auto Scaling

The difference between AWS cloud architecture and the traditional hosting model is that AWS can dynamically scale the web application fleet on demand to handle changes in traffic.

In the traditional hosting model, traffic forecasting models are generally used to provision hosts ahead of projected traffic. In AWS, instances can be provisioned on the fly according to a set of triggers for scaling the fleet out and back in. Amazon Auto Scaling can create capacity groups of servers that can grow or shrink on demand.

## 3.9.3 Key Considerations for Web Hosting in AWS

Following are some of the key considerations for web hosting −

### No physical network devices needed

In AWS, network devices like firewalls, routers, and load-balancers for AWS applications no longer reside on physical devices and are replaced with software solutions.

Multiple options are available to ensure quality software solutions. For load balancing choose Zeus, HAProxy, Nginx, Pound, etc. For establishing a VPN connection choose OpenVPN, OpenSwan, Vyatta, etc.

### No security concerns

AWS provides a more secured model, in which every host is locked down. In Amazon EC2, security groups are designed for each type of host in the architecture, and a large variety of simple and tiered security models can be created to enable minimum access among hosts within your architecture as per requirement.

### Availability of data centers

EC2 instances are easily available at most of the availability zones in AWS region and provides model for deploying your application across data centers for both high availability and reliability.