**7 | Cloud Computing**

**7.1 Indroducción a cloud computing: que es**

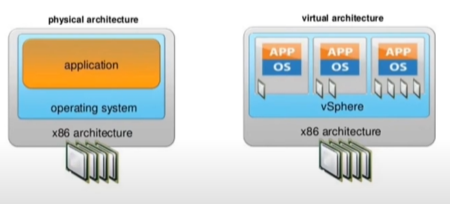
To begin with, we must distinguish between two concepts involved throughout Cloud computing: hardware, which is the set of basic components that make up a server and a Data Center, which is a group of servers that constitute a center of computing resources.

Cloud as a service refers to platforms such as Dropbox, GoogleDrive, OneDrive among others, where a service is offered thanks to the operation of Data Centers through the Internet, so that the user makes use of the computer service without the need for any local requirement on your computer.

In general terms, cloud computing is a system of computing divided into servers, theorized since the 1960s by computer scientists such as John McCarty, who was one of the first to speak of "time-sharing computing", i.e., a computing that would allow multiple users to use the same computer at the same time. However, it was Ramnath Chellappa, in 1997, who first used the term "cloud-computing", referring to a computing paradigm where processing limits are to be mediated solely by economic logic and not by technical limits alone.

Although Jhon McCarty proposed shared computing as a public service, it was not until 2008 that the Cloud service began to be commercialized and it was not until the years 2013-2015 that this new model of resource delivery had an explosive impact on the business habit.

One of the fundamental pillars for Cloud is "Virtualization", which is the ability for multiple virtual machines to connect to the same service without any of the activities of each virtual machine interfering with the activity of the other.



El rol del a virtualización es regular los recursos del servidor aislándolos para dárselos independientemente a cada máquina virtual.

Cloud computing es un “modelo de entrega y consumo de servicios informáticos”, constituye una forma diferente de acceder a unos recursos y capacidades informáticas a través de servidores en un Data Center más no constituye una tecnología en sí. Corresponde a una alternativa al uso de redes locales y computadores personales para el procesamiento y almacenamiento de información.

**7.2 ¿Por qué la nube es necesaria?**

1. Reducir costos de infraestructura, personal y mantenimiento.
2. Desplazar el Hardware.
3. Estandarización de la computación a nivel global y de tamaño de organización.
4. Mayor seguridad informática.
5. Mayor escalabilidad.

**7.3 Principales características de la computación Cloud**

Los sistemas cloud se caracterizan por propiedades como:

1. Seguridad informática: hace referencia a aquellos servicios cloud los cuales ofrecen almacenamiento de datos, donde la información del usuario no solo posee estrictos sistemas de seguridad para asegurar la privacidad de la data, sino que posee restablecimiento de versiones anteriores mediante backups.
2. Accesibilidad: Capacidad inmediata e ilimitada de acceder a un recurso de almacenamiento o procesamiento.
3. Ajuste a la demanda: Corresponde al pago del servicio cloud de acuerdo a la demanda del usuario.
4. Escalabilidad: Es una propiedad de un proceso, red o sistema la cual se define como la flexibilidad del mismo para adaptarse a tamaños de data más y más grandes.
5. Flexibilidad: Las características de los recursos de almacenamiento o procesamiento que se obtienen pueden ser modificadas de manera inmediata.
6. Rapidez: la cantidad de retardos entre la red de servidores es menor que en un computador local.
7. Multiplataforma: Basta tener conexión a Internet para acceder al servicio cloud.
8. Recursos en pool: Sigue el modelo de multiusuario.
9. **Servicio medido:** La utilización de recursos es monitoreada, controlada y relatada de manera anticipada. Esto torna la capacidad de computación esencialmente la misma en relación a un servicio de utilitarios pagados.

**7.4 Como funciona Cloud Computing**

La computación en la nube utiliza una capa de red para conectar los dispositivos de punto periférico de los usuarios, como computadoras, smartphones y accesorios portátiles, a recursos centralizados en la data center.

En principio, un servidor cloud es más seguro que un servidor local ante ataques informáticos o perdidas de información gracias a que estos servicios son usualmente ofrecidos por empresas informáticas enormes que disponen grandes cantidades de recursos humanos y económicos para asegurar el bienestar de su plataforma tecnológica. No obstante, en cuanto al costo de cloud, cloud es más barato en la medida que ahorra al usuario la necesidad de disponer dinero en la construcción de un conglomerado de servidores propios, lo cual involucra personal, instalaciones, métricas de seguridad ambiental y demás factores que no son nada baratos de suplir. No obstante, el servicio cloud puede ser muy costoso en cuanto al costo de la tarifa del servicio como tal, pero si es más barato en términos operativos para las organizaciones.

Los tres gigantes del servicio cloud son:



Pero también, se tienen a:



Dentro de los servicios cloud encontramos:

1. Cómputo: flujos de CPU, RAM y GPU.
2. Almacenamiento: discos duros en máquinas virtuales.
3. Networking: Facilitar la comunicación de la red, la comunicación transparente entre recursos

Oficialmente, Cloud computing es un conjunto de recursos informáticos virtualizados y compartidos para ser utilizados bajo demanda, dentro de su oferta tenemos Servicios de cómputo, almacenamiento y networking que pueden ser desplegados de manera rápida y escalable.

Cloud permite acceso remoto a softwares, almacenamiento y procesamiento de datos por medio de Internet.

**7.5 Tipos de cloud**

**Nube Privada**

Constituida de una sola organización con su propia nube de servidores y software para la utilización. (Sin un punto de acceso público).

**Nube Pública**

Diversas empresas pueden usarla de manera simultánea, pero separadamente. El proveedor de la nube es responsable por el mantenimiento y seguridad.

**Nube Híbrida**

Compuesta por dos o más infraestructuras de nubes distintas que permanecen como entidades únicas, pero que están unidas por una tecnología estandarizada o propietaria.

**Nube Comunitaria**

Diferentes empresas u organizaciones reúnen en pool sus recursos en la nube para resolver un problema común.

**7.6 Ventajas del servicio cloud**

1. **Actualizaciones automáticas de software:** Los proveedores de Cloud Computing son responsables por el mantenimiento del servidor, incluyendo actualizaciones de seguridad.
2. **Capacidad de trabajar en cualquier lugar: el servicio cloud solo depende del acceso a Internet.**
3. **Colaboración:** Todo usuario puede compartir aplicaciones y documentos al mismo tiempo.
4. **Seguridad virtual:** Los datos almacenados en la nube pueden ser accedidos desde cualquier lugar independientemente de la pérdida o daños de uno o más dispositivos.
5. **Flexibilidad/escalabilidad:** Los servicios cloud atienden demandas mayores de manera instantánea.
6. **Recuperación de desastres:** Los proveedores Cloud manejan los problemas de recuperación más rápidamente que las recuperaciones locales.
7. **Menor costo ambiental:** Cloud solo utiliza el espacio necesario en el servidor, reduciendo la huella de carbono de la empresa.

**7.5 Modelos CLoud**

1. **SAAS - Software as a Service:** Es un modelo de cloud donde aplicaciones software son provistas sobre el internet como servicio. En esencia, el proveedor cloud ofrece software en su plataforma.
2. **PAAS – Platform as a Service:** Es un modelo de Cloud que provee una plataforma y un ambiente para desarrolladores para construir, desplegar y manejar aplicaciones. Ofrece un marco de desarrollo completo, incluyendo herramientas, librerías y servicios.
3. **IAAS – Infrastructure as a Service:** Es un modelo de Cloud donde componentes fundamentales de infraestructura de computación son dispuestos al usuario, tales como maquinas virtuales, almacenamiento y networking.

**7.6 DISTRIBUTED DATABASES**

Distributed databases are collections of multiple physically separated databases that communicate through a computer-server network. Some of their advantages and disadvantages are:

1. ADVANTAGES

* Modular development.
* Increased reliability.
* Improved performance.
* Increased availability.
* Quick response time.

1. DISADVANTAGES

* Complex security management.
* Processing complexity.
* More complex data integrity.
* Cost.

Among all the schemes of DDB, in general their types are:

1. HOMOGENEUS: DDB with the same type of DB (SQL, NoSQL), same handler and operating system and data modeling.
2. HETEROGENEOUS: DDB with some of the previous characteristics of different types.

The most common DDB architectures are:

1. CLIENT SERVER: Where there is a main DB and has several DBs that serve as clients or as slaves that will try to obtain data from the main one, to which normally the queries are requested.
2. PEER TO PEER (P2P): Where all points in the DB network are equally important and talk to each other as equals without having to respond to a single entity.
3. MULTI-DATABASE MANAGER: Is a DB conformed for multiple types of DDB architectures.

Whenever a DB is stablished, there are two types of design strategy:

1. TOP DOWN: When a clint or organization plan the entire architecture of the DB such that it configures from the top bottom according to the organization needs and try to select the better networking schemes together with the better IT resources.
2. BOOTOM UP: Is a generated DB from already existing DBs such that the organization tryes to build on top of it.

Whenever we talk about data distribution, there are three types:

1. HORIZONTAL FRAGMENTATION: (Sharding) Is the splitting of the SQL DB by rows.
2. VERTICAL FRAGMENTATION: Is the splitting of the SQL DB by columns.
3. MIXED FRAGMENTATION: Is the splitting of the SQL DB by columns and rows.

Whenever we talk about data replication, there are three types:

1. Full replication.
2. Partial replication.
3. No replication.

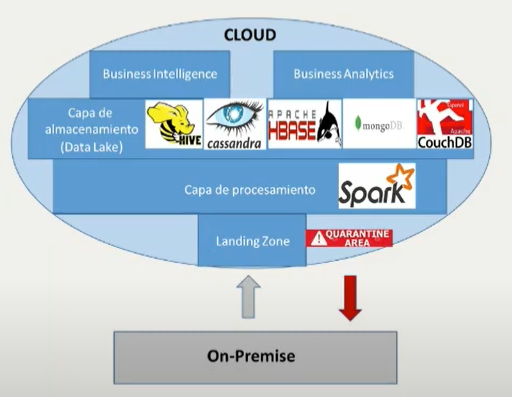
Whenever we talk about data distribution, there are three types:

1. Centralized.
2. Partitioned.
3. Replicated.

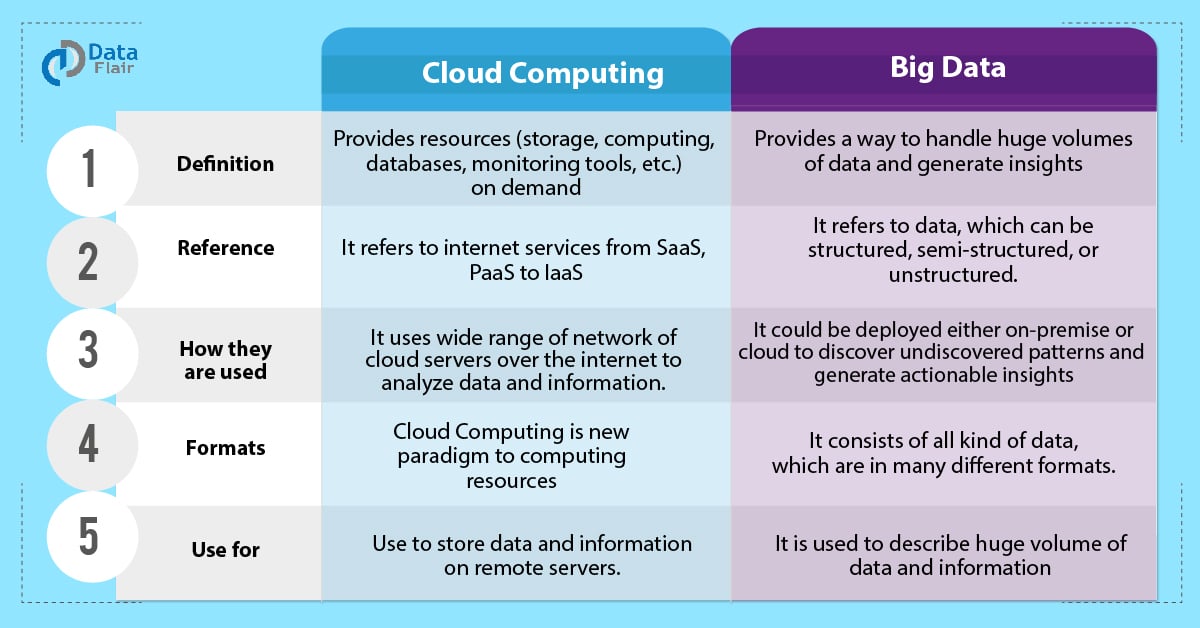
**7.6 Soluciones para BigData en la nube**

Las bases de datos *On-Premise* hacen referencia a todas a quellas bases de datos almacenadas en medios locales. Así, en especial la BigData, requiere de Cloud para su procesamiento, almacenamiento y networking. En términos generales, las estructuras de Cloud cuentan con tres capas, la cual constituye un servicio diferente dentro de la arquitectura del servicio Cloud provisto:

1. Landing Zone: es la zona de aterrizaje primera de los datos, allí, estos son testeados mediante criterios de calidad general y acordes a las necesidades de la organización para asegurarse de la viabilidad de la data en etapas superiores. En caso de no cumplir con los criterios de calidad, los datos son transferidos a una zona de cuarentena donde será procesados, arreglados o devueltos al servidor on-premise para su reformulación.
2. Capa de procesamiento: En la capa de procesamiento (en donde Spark es una API gestionada desde Python, scala entre otras, es una de las más utilizadas en la industria) donde se llevan a cabo procesos como los enlistados a continuación. En general, la capa de procesamiento se emplea para hacer analítica de datos, ciencia de datos y estructuración de datos.
   1. Batch Processing: procesamiento de datos en paralelo con base a lenguajes de programación como Python, R, Java y otros.
   2. Real-time Stream Processing: procesamiento y análisis de datos en tiempo real.
   3. Machine Learning: Formulación de modelos predictivos mediante aprendizaje automático.
   4. Interactive Data Analysis: interacción tipo query y análisis de data estructurada tipo SQL.
   5. Graph Processing: procesamiento de la data en forma de grafo.
   6. Data ETL (Extract, Transform, Load)
3. Data Lake: Corresponde a un conjunto de tecnologías en donde la data procesada y estructurada va a residir.



El **Big Data** está compuesto por toda clase de información que tienen las empresas de los usuarios. Esta ha sido **obtenida durante su navegación** por Internet, con el uso de webs y aplicaciones en línea. Así pues, dada la gran cantidad de almacenamiento que supone toda esta información, el mejor sitio para almacenarla es en la nube, mediante el**Cloud Computing**.



BigData and Cloud Computing are two entangled concepts, in general, BigData is about managing and analyzing large and complex datasets, while Cloud Computing is about delivering computing resources and services over the internet. While they are distinct conepts, they often intersect, as Cloud Computing provides a scalable and cost-effective infrastructure for processing BigData workloads. We can list the following main differences

1. Definition and Focus:

* *BigData:* BigData refers to large volumes of data that cannot be effectively managed, processed, and analyzed using traditional data processing techniques. It focuses on extracting insights and value from massive and diverse datasets, oftedn characterized by their Volume, Velocity and Vairety.
* *Cloud Computing:* Is a computing paradigm that involves the delivery of computing resources, including storage, processing power, and applications over the internet. It focuses on providing on-demand Access to computing resources, scalability, and flexibility in a cost-effective manner.

1. Purpose and Use cases:

* *BigData:* Is primarly concerned with extracting insights, patterns, and trends from large and complex datasets. It involves techniques such as data mining, machin learning, and analytics to derive value and make data-driven decisions.
* *Cloud Computing:* Focuses on providing computing resources and services to users and organizations. It offers on-demand Access to virtualized infrastructure, platforms, and software applications.

1. Technology and Infrastructure:

* *BigData*: Involves technologies and frameworks that enable the storage, processing, and análisis of large datasets. This includes distributed file systems as Hadoop Distributed File System (HDFS), DISTRIBUTED PROCESSING FRAMEWORKS LIKE Apache Spark and Apache Hadoop, and specialized databases like Apache Cassandra and Apache Hbase.
* *Cloud Computing*: Relies on virtualization technologies and infrastructure management to deliver computing resources as services. It involves virtual machines, containers, and serverless computing models. Cloud providers, such as AWS, Microsoft Azure and GCP, offer infrastructure services like virtual machines, storage and networking.

1. Data vs Computing Resources:

* *BigData:* Focuses on the data itself, including storage, processing, and análisis techniques to derive insights from the data. It emphasizes the management, processing, and utilization of large diverse datasets.
* *Cloud Computing:* Focuses on delivering computing resources as a service, enabling users to Access and utilize computing power, storage, and software applications over the internet. It emphasizes the provision and management of computing resources.

Before going further, lets define what is data warehouse. A data warehouse is a large, centralized repository of integrated data from multiple sources within an organization. It is designed to support business intelligence (BI) activities such as reporting, analytics, and decision-making.

Some key characteristics and components of data warehouse are:

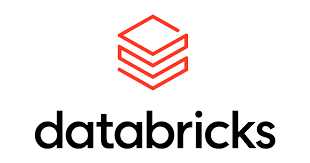
1. **Integration:** Data warehouse integrate data from various sources, such as transactional databases, spredsheets, and external systems. The data is transformed and standardized to ensure consistency and compatibility.
2. **Subject-oriented:** Data warehouses are organized around specific subject areas or domain of interest to the organization, such as sales, customer data, or interventory. They provide a consolidated view of data related to these subjects.
3. **Time variant:** Data warehouse store historical data, allowing analysis and reporting over different time periods.
4. **Non-volatile:**  Data in a warehouse is typically read-only and not directly updated or modified. Instead, data updates are performed in the source systems, and periodic extract, transform, and load (ETL) processes are used to update the data warehouse.
5. **Structure and schema:** Data warehouse employ a structured schema, such as a star schema or snowflake schema, to organize data into dimensions and fact tables. This schema design facilitates efficient querying and analysis.
6. **Analytics and reporting:** Data warehouses provide a foundation for business intelligence and analytics activities. They support complex queries, ad-hoc reporting, online analytical processing (OLAP), and data mining to extract insights and support decision-making.
7. **Performance optimization:** Data warehouses employ various techniques to optimize query performance, such as indexing, partitioning, and aggregation. These techniques help ensure that analytical queries are executed efficiently, even on large volumes of data.

Data warehousing and BigData are closely related but distinct concepts. Here´s an explanation of their relationship:

1. **Data volume:** One of the main differences between traditional data warehousing and Big Data is the volume of data they handle. Data warehouses typically store and process structured data from various sources, but their scale is limited compared to Big Data. Big Data, on the other hand, deals with massive volumes of both structured and unstructured data, often generated in real-time or at high velocity.
2. **Data Variety:** Data warehouses primarily handle structured data with a predefined schema. They integrate data from different sources but typically follow a consistent structure. In contrast, Big Data encompasses various data types, including structured, semi-structured, and unstructured data, such as text, images, videos, social media feeds, sensor data, and more.
3. **Data processing:** Data warehousing focuses on aggregating and summarizing data to support business intelligence and reporting. It employs SQL-based querying and follows a predefined schema for analysis. Big Data, on the other hand, involves processing vast amounts of data using technologies like Hadoop and Spark. It often involves distributed processing, parallel computing, and complex analytics algorithms to derive insights from diverse and large-scale datasets.
4. **Scalability and Flexibility:** Data warehouses are designed for scalability but operate within defined hardware and storage limits. Big Data, by contrast, emphasizes horizontal scalability, enabling organizations to scale their data processing and storage capabilities dynamically as data volumes grow. Big Data technologies like distributed file systems and NoSQL databases provide the flexibility needed to handle the ever-increasing data requirements.
5. **Integration:** In some cases, organizations incorporate Big Data technologies and frameworks into their data warehousing architecture to handle the processing and analysis of large-scale, diverse datasets. This combination allows for enhanced analytics capabilities by leveraging both structured and unstructured data sources.

Overall, while data warehousing focuses on structured data and provides a structured, centralized view for analysis, Big Data encompasses large volumes, diverse types, and complex processing requirements. Organizations often combine the strengths of both approaches to enable comprehensive data analysis and decision-making in the era of Big Data.

In terms of PaaS services, **Databricks** is another example of BigData Cloud solution that focuses primarily on offering the platform service for this type of computing, however, Databricks does not own the computing power as such, this platform uses Apache Spark as backend.



Dentro de las soluciones Big Data más comunes open source que constituyen un estándar en la industria, encontramos:

1. Hadoop.
2. Apache Spark.
3. MongoDB
4. ElasticSearch
5. ApacheStorm

**HADOOP**



Hadoop es un sistema de computo BigData open source considerado el framework estándar para este tipo de soluciones. En él, se pueden almacenar, procesar y analizar grandes cantidades de datos (usada por empresas como Facebook y Yahoo), puede soportar múltiples OS y se usa frecuentemente en conjunto con las principales nubes tales como Amazon EC2/S3 o Google Cloud.

Desde los años 1990 y principios de los 2000, se crearon buscadores (o motores de búsqueda) e índices para ayudar a localizar información relevante dentro de contenido basado en texto, el WWW al tener un crecimiento explosivo, demando la creación de buscadores en línea que permitieran acceder a la información de manera fácil y rápida. Uno de estos proyectos fue un buscador Web de código abierto llamado **Nutch** (Apache Nutch Proyect) cuya idea original fue de Doug Cutting y Mike Cafarella quienes d eseaban generar resultados de búsquedas en la Web a mayor velocidad distribuyendo datos y cálculos en diferentes computadoras de modo que se pudieran procesar múltiples tareas de manera simultánea (en estos años el proyecto de buscador de Google andaba paralelamente en desarrollo). Sin embargo, el proyecto no se hizo viable por cuestiones de costos.

En 2003, Google lanzó un paper científico llamado GFS (Google File System), en donde explicaban la solución que determinó Google para el almacenamiento de enormes cantidades de información en una forma distribuida, una de las soluciones que el proyecto Apache Nutch nesitaba para reducir costos en su sistema distribuido. En 2004, Apache Nutch aplica otra de las soluciones expuestas por Google llamado el sistema Map-reduce lanzada al público científico en 2004.

Combinando las ideas de Google, en 2005 ANP empezó con tan solo 20-40 clusters donde solo Doug Cutting y Mike Cafarella trabajaban dentro del proyecto. Debido a que necesitaban más capacidad de cómputo y mano de obra, Yahoo, una de las primeras compañías en incursionar en los servicios de búsqueda, se interesó en el proyecto ANP y decidió hacer una alianza donde Yahoo proveía servicios de mano de obra a ANP.

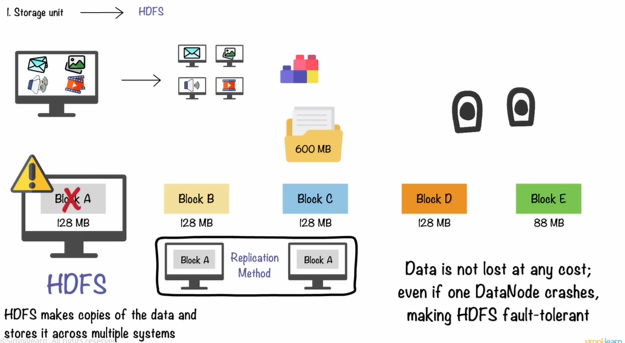
En 2006, Cutting oficialmente se unió a Yahoo con la idea de desarrollar un sistema de almacenamiento rentable, open source y altamente escalable llevandose con él el proyecto Nutch, así como también ideas basadas en los trabajos iniciales de Google con la automatización del almacenaje y procesamiento de datos distribuidos. El proyecto Nutch fue dividido – la parte del rastreador Web se mantuvo como Nutch y la parte de cómputo y procesamiento distribuido se convirtió en Hadoop. Hadoop recibió este nombre gracias a que Cutting tenía un hijo cuyo juguete favorito era un elefante llamado Hadoop.

En 2007, Hadoop fue probado por primera vez en 1000 servidores, para que, en 2008, Yahoo presentara Hadoop como proyecto de código abierto. Hoy día, la estructura y el ecosistema de tecnologías de Hadoop son gestionados y mantenidos por la Apache Software Foundation (ASF) sin fines de lucro, que es una comunidad global de programadores de software y otros contribuyentes. Foundation (ASF) puso Hadoop a disposición del público en noviembre de 2012 como Apache Hadoop.

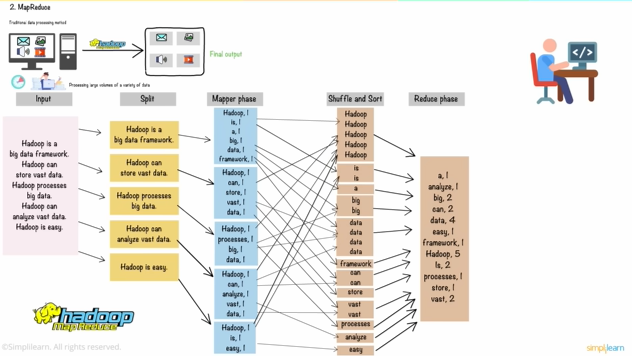
Hadoop emplea modelos de programación distribuida para el almacenamiento y procesamiento de información. De manera general, dispone de tres tipos de servicios o ecosistemas.

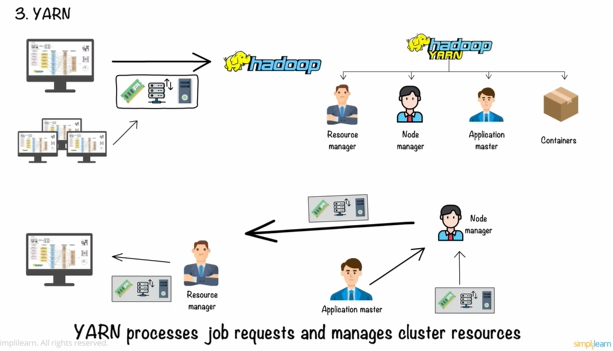
Basado en el sistema Map-Reduce, HDFS, se encarga de particionar la data en bloques iguales y almacenarlos de manera redundante en diferentes instancias de servidor, de modo tal de que la información sea recuperable.



Una vez la data ha sido almacenada, sebe ser procesada para salvar más recursos de almacenamiento y para facilitar su accesibilidad. Para ello se emplea Map-Reduce, proceso el cual consiste en una segmentación en partes iguales de la data en cada bloque, para posteriormente ejecutar una tokenización en cada bloque asignando un número a cada elemento. Luego, se agrupan los elementos similares y se suman los conteos de los elementos encontrados en la fase de toknizacion y se ordenan. Finalmente se consolida el resultado mediante un código de programación que sumarice los resultados de la transformación como una data estado-cantidad de este estao.



Una vez el mapeo-reducción ha sido ejectutado, es posible correr la data dentro del cluster de BigData deHadoop, para por ejemplo, su análisis. Hadoop Yarn constituye el distribuidor de recursos de Hadoop, regulando la cantidad de almacenamiento, ram y flujo de internet que la tarea de procesamiento requiere.



El ecosistema Hadoop, posee varias herramientas de procesamiento BigData tales como:



**APACHE SPARK**

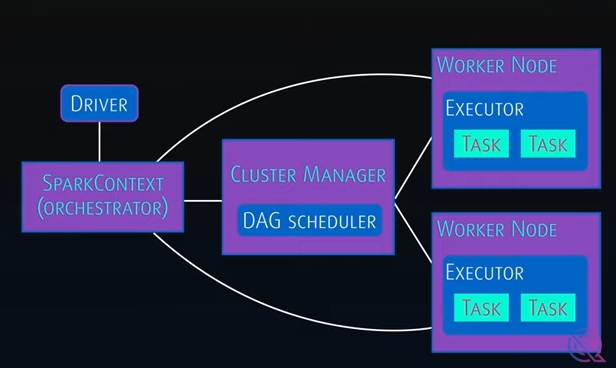
****

Spark fue desarrollado en 2009 en el Berkeley Lab por Matei Zaharia para ser abierto al público como recurso open source en 2010. En 2013, el control de Spark fue cedido a la fundación Apache para en 2014 ser designado como un proyecto informático de alto nivel. Spark fue desarrllado como un proyecto BigData de rápido funcionamiento y alto escalamiento que pudiera lidiar con los inconvenientes e ineficiencias de Hadoop Map-Reduce.

La tecnología de Spark está basada en el RDD (Resilient, Distributed Dataset), la cual es una abstracción que representa una colección de read-only objects (objetos de lectura única) distribuida a lo largo de un cluster de computadoras. El RDD puede usarse a partir de tablas de bases de datos, bases de datos SQL, bases de datos NoSQL, bases de dato HDFS Cloud Storage entre otras.

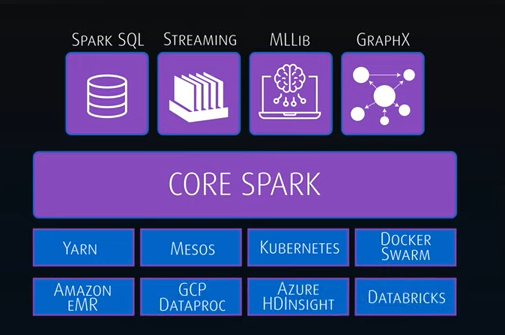
RDD permite el uso de funciones Map-Reduce estándar en conjunto con unión, filtrado y agregación de bases de datos. Una de las principales diferencias entre Hadoop y Spark, es que el primero requiere que sus procesos sean llevados a cabo en memoria de disco duro mientras que el seguno, ejecuta todas sus actividades en RAM.

* OLAP: It is an online data retrieval and data analysis system.
* OLTP: It is an online transactional system and manages the modification of the database.

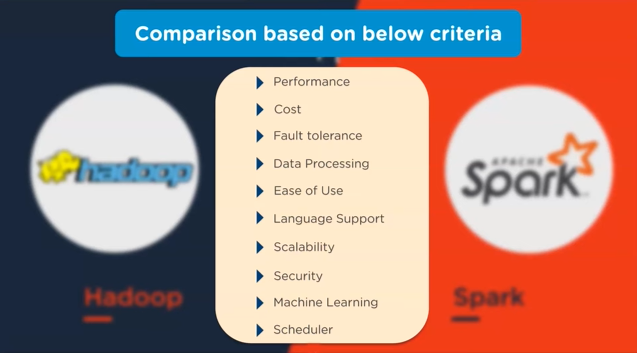


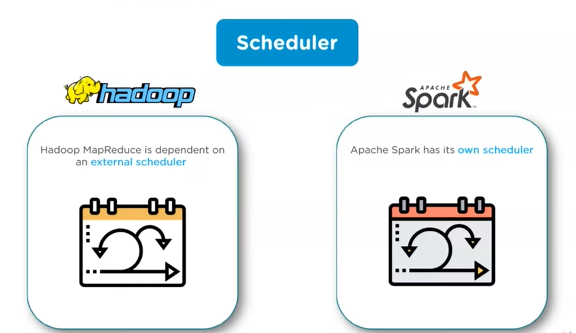
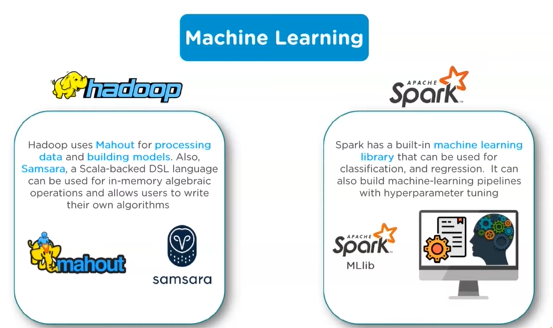
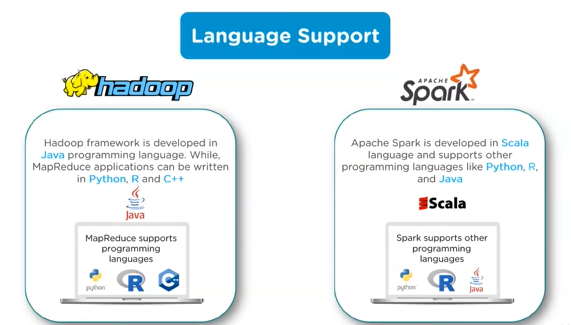
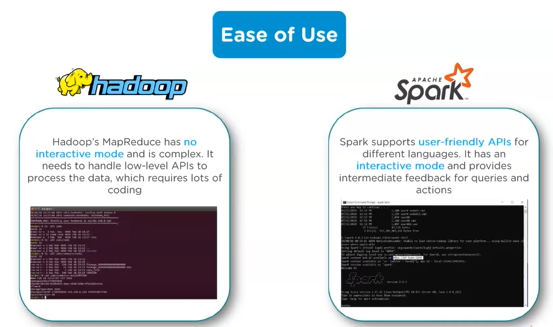
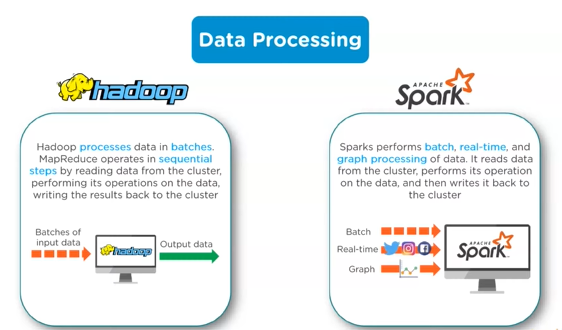
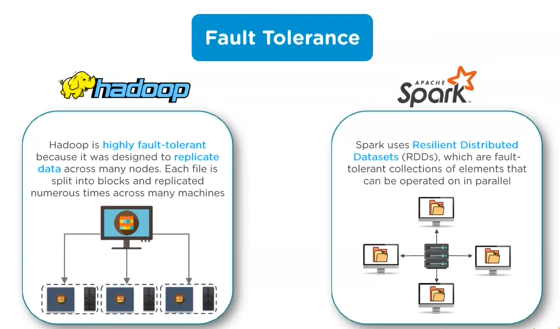
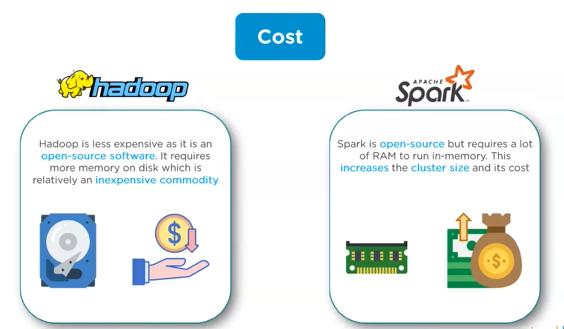
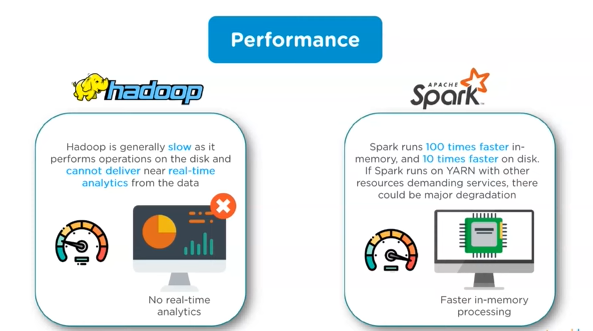
Spark permite trabajar cómodamente con lenguajes como Python, Java, R, scala entre otros. Spark posee cuatro compoentes fundamentales, Skarp SQL el cual procesa datos estructurados con lenguajes de programación tradicionales, Spark Streaming, la cual es una capa de procesamiento de Skarp por bloque para el procesamiento y análisis en tiempo real de datos tipo streaming, Spark MLLIB, la cual es la capa de Machine Learning de Skpark, y finalmente, Spark GraphX, la cual es la capa de procesamiento de Grafo por bloque de este ambiente.

De manera general, puede pensarse a Hadoop como una capa de almacenamiento BigData mientras a Spark como la capa de procesamiento BigData construida sobre Hadoop.



De manera general, cuando se habla de soluciones BigData se tienen en cuenta los siguientes aspectos, a continuación, se compararán entre Hadoop y Spark.



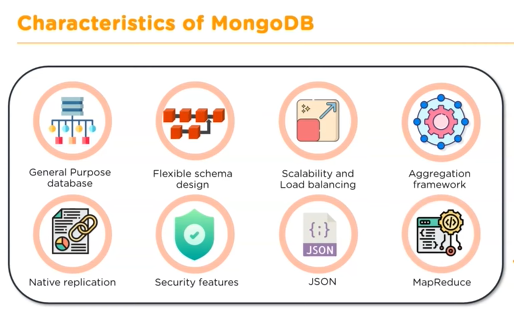
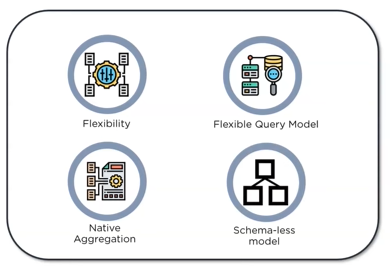


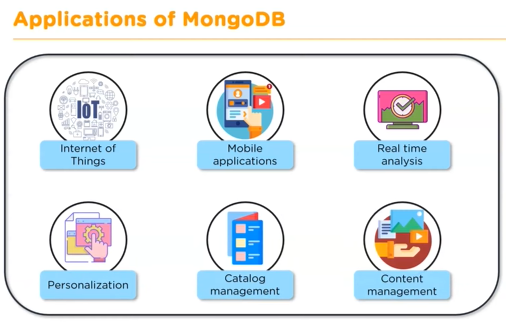
**MONGODB**

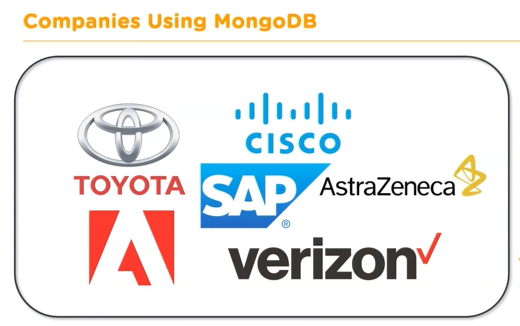


MongoDb is currently the most popular NoSQL-oriented database specialized in document storage. It was created in 2007 by the company 10gen, thanks to the fact that DoubleClick, a company that regulated thousands of ads per second, began to have problems of speed and scalability using traditional databases at the time. This encouraged them to create their own database project based on non-relational data such as BJSON and JSON.

Their approach allows that by not having such a strict structure as in SQL databases, documents which are stored as collections, can be called through queries quickly and easily. By being able to store documents in a flexible and compact way, it avoids the need to perform complicated database union operations or union for the assembly of file components required in a query, which greatly facilitates the horizontal scalability of the database not only because of the above, but also because being able to store documents as a collection, allows migrating, calling and compactly storing groups of documents.







**ELASTIC SEARCH**



ElasticSearch is a JSON-based NoSQL database. Elasticsearch is a powerful tool for searching large amounts of data, especially when the data is of a complex type.

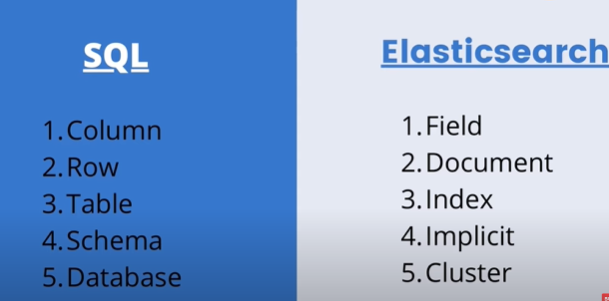
data, especially when the data is of a complex type. It allows us to index and analyze in real time a large volume of data and make queries on it. An example of use are full-text queries.

full-text queries; since the data is indexed, the results are obtained very quickly. digital. Unlike other similar systems, it does not need to declare a schema of the information we add, we do not know exactly what form the data will take. With Elasticsearch we can do complicated text searches, visualize the state of our nodes and scale without and scale without too much need, should we need more power. more power.

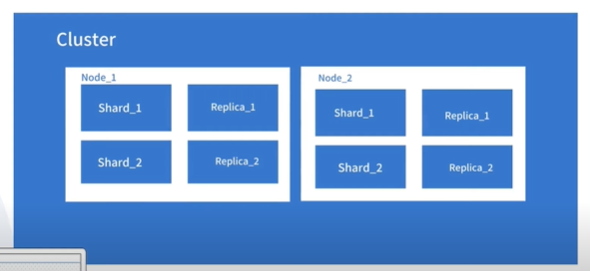
* Scalable open-source, multitenant-capable, full-text, distributed search and analytics engine.
* Provide real time data analytics and searches.
* Document oriented database.

Some primary use cases are:

* Application search.
* Website search.
* Enterprise search.
* Logging and Log analytics.
* Infrastructure metrics and container monitoring.
* Security analytics.
* Business analytics.







**8 | Fundamentals of Data Bases and Relational Data Bases**

**8.1 Introduction to Data Bases**

A database is a collection of information that is systematically stored and queried. A Database Management System is the software in charge of administering, managing and creating databases. From it we can write, create the structure, query, ensure the integrity and control the access to our databases. Although databases are a tool of human memory that can be traced back to the beginning of the species, let's talk about this topic from the 1940s onwards.

Digital computers were formally invented in the 1940s, but despite being an excellent computing tool, they lacked memory, so it was not possible to store databases digitally at that time. In the 50's they began to store data on magnetic tapes, however they are sequential storage mechanisms that involve going through the entire tape to the point where we get to the desired information to consult. This first way of storing databases is called "browsing databases".

In the 70's, Edgar Frank Codd wrote a document where he stipulated the basis for the creation of the relational model of databases, which were much better than the navigational databases not only because of the implicit mathematical model but also because they worked on hard disks. Relational databases completely change the world market. The most famous data management model in the 70's was ORACLE, a SQL-like data management system founded by Larry Ellison. Later in the 1980s Microsoft SQL Server appeared, while open source SQL data systems such as MySQL and PosgresSQL appeared in the 1990s. However, years later, Microsoft bought SunMicrosystems, taking Java and MySQL into its own pocket.

In the 2000s with the emergence of Web 2.0, with the democratization of the Internet and allowing people to decide the destination, format and type of content on the web, a huge amount of information begins to be stored on the Internet. It is because of this ridiculous amount of information that SQL databases begin to lack effectiveness due to their limited scalability.

Before going further, lets lists the various types of databases:

* **Distributed databases:** A distributed database is a type of database that has contributions from the common database and information captured by local computers. In this type of database system, the data is not in one place and is distributed at various organizations.
* **Relational databases:** This type of database defines database relationships in the for of tables. I t is also called relational database RDBMS. Some examples of RDBMS systems include MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.
* **Object-oriented databases:** This type of computers supports the storage of all data types. The data is stored in the form of objects. The objects to be held in the database have attributes and methods that define what to do with the data. PostgreSQL is an example of an object-oriented relational DBMS.
* **Centralized database:** It is a centralized location, and users from different backgrounds can access this data. This type of computers databases store application procedures that help users access the data even from a remote location.
* **Open source-databases:** This kind of database stored information related to operations. It is mainly used in the field of marketing, employee relations, customer service among others.
* **Cloud databases:** Database build on a virtualized environment, it is a database that operates over cloud computering-based servers.
* **Data warehouses:** Is an information system that contains historical and commutative data from single or multiple sources or organizations. It is characterized by being a very large amount of information.
* **NoSQL databases:** It is a database used for humongous sets of distributed data. It is characterized by having no relational storage scheme and a flexible organization of the data.
* **Graph databases:** A graph-oriented database uses graph theory to store, map, and query relationships. These kinds of computer databases are mostly used for analyzing interconnections and direct relationships between agents.
* **Personal databases:** A personal database is used to store data stored on personal computers that are smaller and easily manageable. The data is mostly used by the same department of the company and is accessed by a small group of people.
* **Document/JSON database:** Document oriented database where the data is kept in a document collection, usually using the XML, JSON, BSON formats.
* **Hierarchical databases:** This type of DBMS employs the “parent-child” relationship of storing data. Its structure is like a tree with nodes representing records and branches representing fields.
* **Network DBMS:** This type of DBMS supports many-to-many relations. It usually results in complex database structures. RDM Server is an example of database management system that implements the network model.

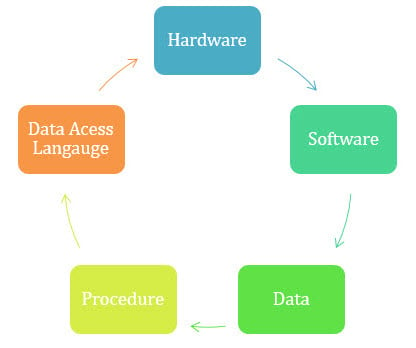
**8.2 Database Management Systems (DBMS)**

**Database Management System (DBMS)** is a collection of programs that enable its users to access databases, manipulate data, report, and represent data. It also helps to control access to the database. Database Management Systems are not a new concept and, as such, had been first implemented in the 1960s.

Charles Bachman’s Integrated Data Store (IDS) is said to be the first DBMS in history. With time database, technologies evolved a lot, while usage and expected functionalities of databases increased immensely.

* **Hardware:** Physical, electronic devices like computers, I/O devices, storage devices, etc., that offers the interface between computers and real-world systems.
* **Software:** Set of programs used to manage and control the overall database. This includes the database software itself, the OS, the network software used to share the data among users, and the application programs for accessing data in the database.
* **Data:** Raw and unorganized facts that requires to be processed to make it meaningful. Data can be simple or complex information, objects or metadata about objects. Between data examples we find objects, facts, observations, perceptions, numbers, characters, symbols, images, etc.
* **Database Access Language:** Structured informatic language used to access, query, update or retrieve data.
* **Procedure:** Set of instructions and rules that help you to use the DBMS. It is designing and running the database using documented methods, which allows you to guide the users who operate and manage it.

In this order of ideas, the database components are:



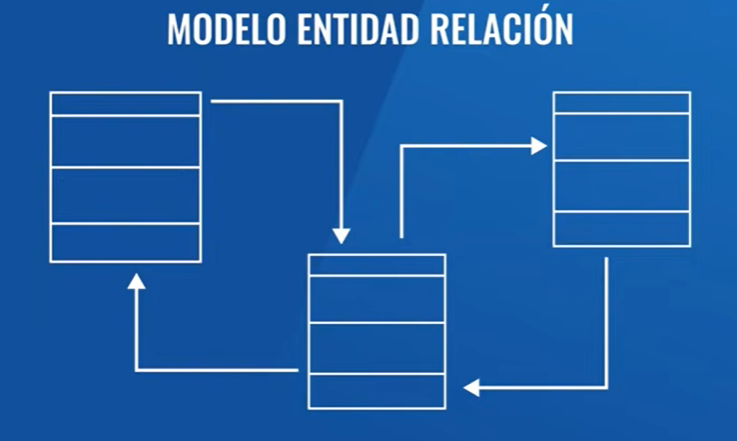
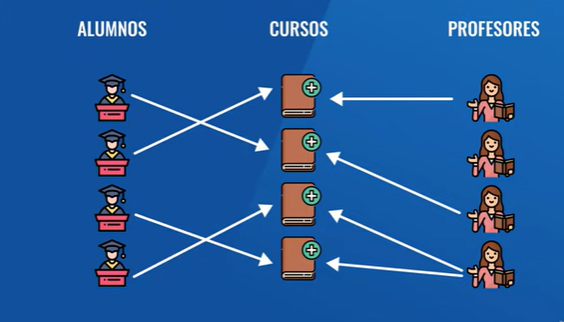
Some important historical records to have in mind about databases are:

* **1960 –** The concept of databases emerged, and the term “Data Management System” was introduced.
* **1960** - Charles Bachman designed the first DBMS system.
* **1964** – IBM introduced the IBM System/360 mainframe computer, which included the first commercially available hierarchical database management system called IMS (Information Management System).
* **1970** - Edgar Frank Codd introduces the theoretical foundations about RDB (Relational Data Bases) in the paper titled “*A Relational Model of Data for Large Shared Data Banks”.*
* **1974** – IBM developed System R, a research project that implemented the relational model and introduced the SQL (Structured Query Language) as the standard query language for relational databases.
* **1976 -** **Peter** Chen coined and defined the Entity-relationship model also known as the ER model.
* **1977** – Oracle Corporation (originally called Software Development Laboratories) was founded, and the Oracle Database, bases on the relational model, was introduced successfully in 1979.
* **1979 –** Larry Ellison, Bob Miner, and Ed Oates and their company Software Development Laboratories (SDL) later renamed to Oracle Corporation, launch their database Oracle which successfully implemented the SQL language.
* **1985 –** Object-oriented DBMS develops, combining object-oriented programming principles with database management systems.
* **1985** – Microsoft released its first version of SQL Server, a relational database management system.
* **1990 –** Microsoft ships MS access, a personal DBMS that displaces all other personal DBMS products.
* **1995 –** First Internet database applications.
* **1997** – XML is applied to database processing. Many vendors begin to integrate XML into DBMS products.
* **2000** – The term “NoSQL” (Not Only SQL) was coined to describe a new wave of non-relational databases that aimed to address scalability and performance challenges posed by BigData and web applications.
* **2003** **–** The term “NeSQL” was introduced to describe a group of modern relational databases that aimed to combine the scalability of NoSQL with the ACID properties of traditional SQL databases.
* **2010**­ – Apache Cassandra, a highly scalable and distributed NoSQL database, became open source and gained popularity for handling BigData workloads.
* **2012** – MongoDB, a document-oriented NoSQL database, gained significant traction and popularity due to its flexibility and ease of use.
* **2016** – Google introduces Spanner, a globally distributed relational database system, designed to provide strong consitstency, scalability, and high availability across multiple regions.

**8.3 Relational SQL databases**

Simply put, a relational database system is based on the entity-relational database model which is a system of relationships between structured data tables through the conjunction of object records and the attributes (fields/columns) that make the objects what they are. In a SQL type database, the database design is established by distinguishing the attributes of the objects and the logical structures that relate the tables to each other.

In this way, a single database, when partitioned into multiple databases or tables, allows a more consistent way of handling information, avoiding redundancy of fields and records and allowing queries to be more easily customized by means of set theory.



As an example, let's imagine a university database. In a university there will be three types of objects or entities that make up the academic activity: students, courses and professors. In a SQL database a table is generated for each entity where each record or row will represent one of the possible instances of each object, where each object (students, courses, students) is defined by a series of attributes or columns. In the SQL database the information of each type of object is stored in a table destined for that type of object so that a query involving information about all or some of the types of objects in the database will require relating the information in the tables.

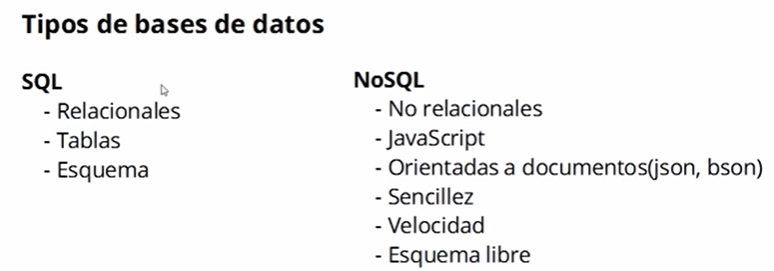
For example, if we want to know the records of those students who receive the course A, B and C given by the professors P1 and P2, we must establish a logical relation of the form "Students who attend courses (A,B,C) given by (P1,P2)", a relation formally established by means of the SQL or Structured Query Language.

*Normalization* in a SQL database refers to the process of organizing and structuring data in a relational database to minimize redundancy, improve data integrity, and optimize database performance. It involves breaking down a database into a multiple table and establishing relationships between them based on their functional dependencies.

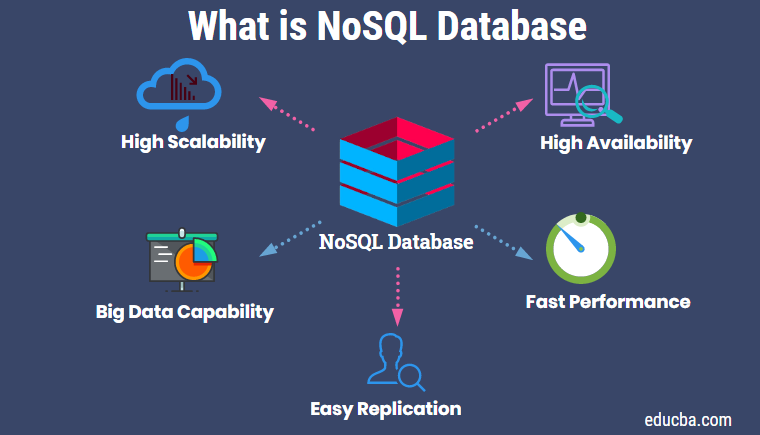
**9 | Non-Relational Data Bases**

**9.1 Introduction to NoSQL databases**

NoSQL databases are non-relational Data Management Systems that does not require a fixed schema. They possess a lack of joins and high scalability. The major purpose of using NoSQL databases is for distributed data stores with humongous data storage needs. Specially, NoSQL is used for BigData and real-time web apps. NoSQL stands for “*Not Only SQL”*. The NoSQL concept was introduced by Carl Strozz in 1998.



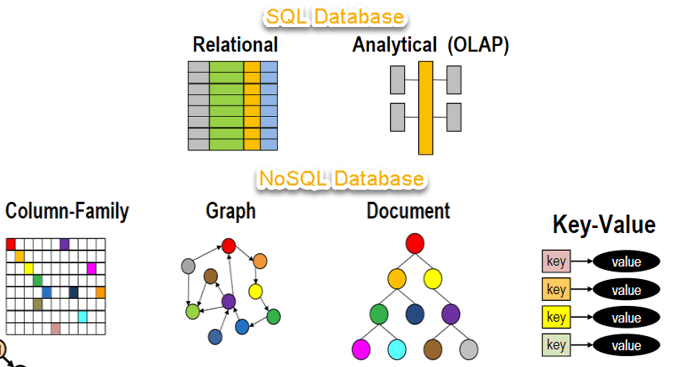
Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data. Some of the most remarkable qualities of NoSQL databases are:



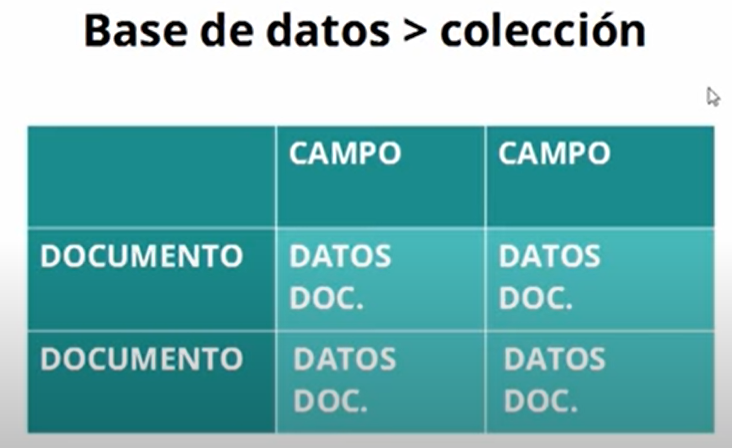
Some of the differences between SQL databases and NoSQL databases are:

|  |  |  |
| --- | --- | --- |
| **Property** | **SQL Databases** | **NoSQL Databases** |
| **Data Model** | Relational data model (tables with rows and columns) | Flexible data model (key-value, document, columnar, graph, etc.) |
| **Schema** | Structured schema defined before data insertion. | Dynamic schema with no strict schema requirements. |
| **Scalability** | Vertical scalability. | Horizontal scalability. |
| **Query Language** | SQL | Various query languages depending on the NoSQL database type. |
| **ACID Compliance** | Generally, ACID-compliant (Atomicity, Consistency, Isolation, Durability) | May or may not be ACID-compliant, depending on the specific NoSQL database. NoSQL are BASE suitable. |
| **Scalability Flexibility** | Limited scalability due to the relational structure. | High scalable, suitable for distributed systems. |
| **Data Integrity** | Strong data consistency and integrity enforced by constraints. | May sacrifice consistency for availability and partition tolerance. |
| **Flexibility** | Less flexible for handling unstructured or changing data. | More flexible for handling unstructured or evolving data. |
| **Use Cases** | Traditional enterprise applications. | BigData, real-time web applications, and high scalable systems. |

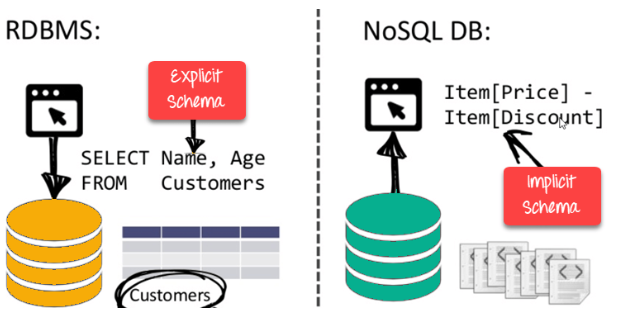
Some of the main types of SQL and NoSQL databases schemes are:



Where between a SQL and a NoSQL database, generally speaking, we can stablish the following parallelism:

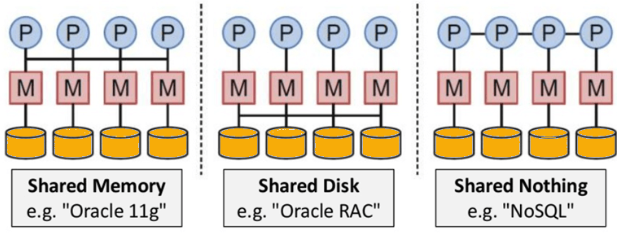


|  |  |
| --- | --- |
| SQL | NoSQL |
| Database | Collection |
| Row | Documents |
| Column | Fields |
| Registers (data cells) | Data of the Document |



NoSQL databases never follow the relational model; therefore, they never provide tables with flat fixed-column records and they doesn´t require object-relational mapping and data normalization. As they do not possess explicit schema, they are sayed to be “schema free” or with “implicit schema”. Most of the APIs protocols are mostly used with HTTP REST with JSON, but they lack of a standard query language as in the case of SQL databases. Since NoSQL databases are primarily intended for high-performance uses, each NoSQL DBMS vendor is prepared to offer distributed data management.

* They offer horizontal auto-scaling and fail-over capabilities.
* Often ACID concept can be scarified for scalability and throughput.
* Mostly no synchronous replication between distributed nodes.
* Commonly Shared Nothing Architecture.



**9.2 Types of NoSQL Databases and characteristics**

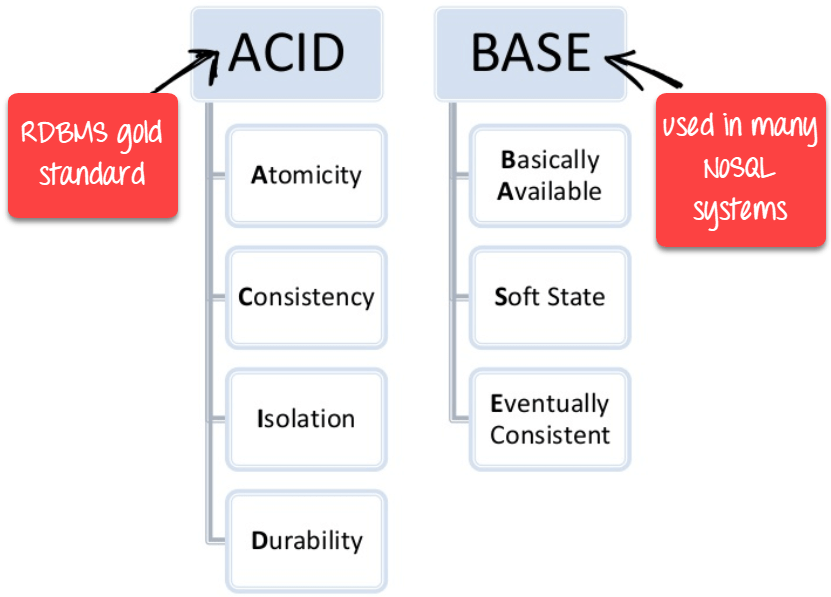
NoSQL databases are mainly categorized into four types: Key-value pair, Column-Oriented, Graph-based, and Document-oriented. Every category has its unique attributes and limitations. None of the above is better to solve all the problems, then, users should select the database based on their product needs.

Within NoSQL databases there is something called the CAP theorem, which states that is impossible for a distributed data store to offer more than two out of three guarantees:

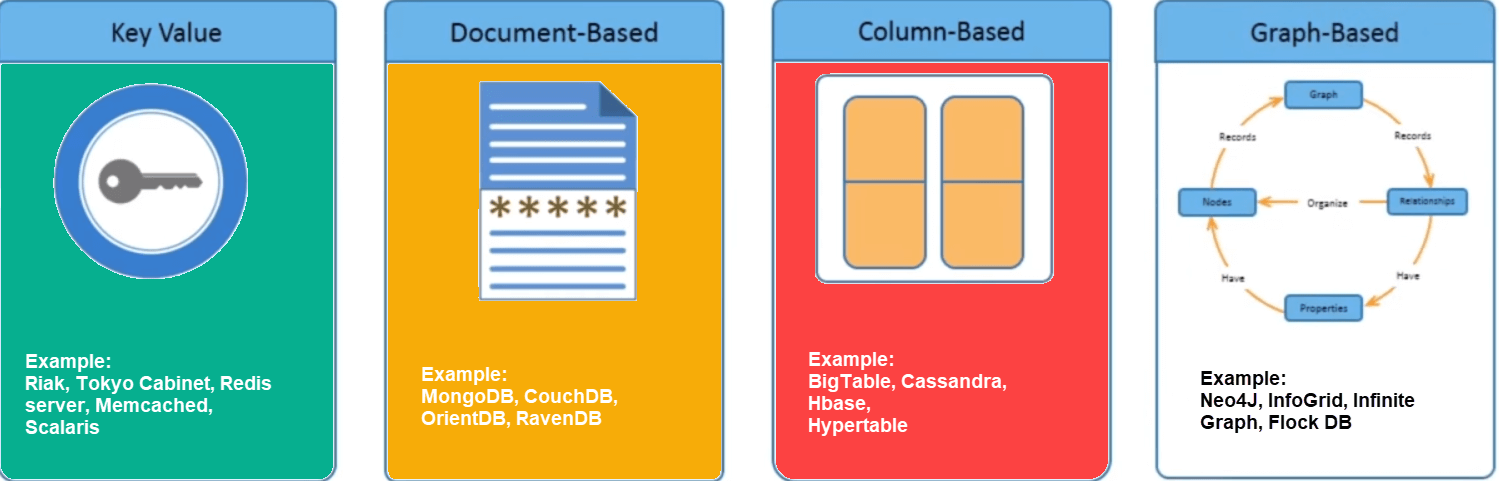
1. **Consistency:** The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.
2. **Availability:** The databases should always be available and responsive. It should not have any downtime.
3. **Partition Tolerance:** Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

**Eventual Consistency** means to have copies of data on multiple machines to get high availability and scalability. Thus, changes made to any data item on one machine has to be propagated to other replicas. Data replication may not be instantaneous as some copies will be updated immediately while others in due course of time. These copies may be mutually, but in due course of time, they become consistent. Hence, the name eventual consistency. This is another fundamental characteristic of NoSQL database in comparison to SQL databases, as SQL has ACID compliance while NoSQL has BASE compliance (Basically Available, Soft state, Eventual consistency).

* Basically, Available means that the DB is available all the time based on CAP theorem. (availability mostly related)
* Soft state means that even without an input, the system state may change. (consistency related)
* Eventual consistency means that the system will become consistent over time. (partition tolerance and update mostly related)

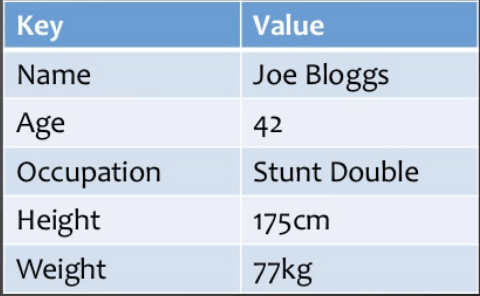


The main categories of NoSQL databases types are:



**KEY VALUE PAIR BASED**

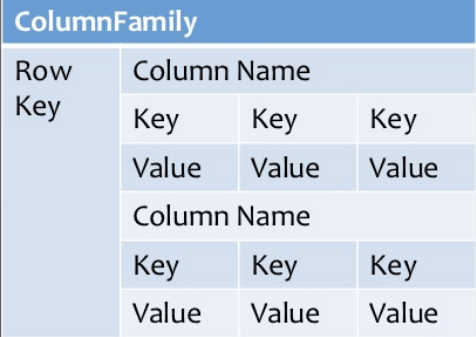
Data is stored in key/value pairs, it is designed in such a way to handle lots of data and heavy load. Key-value pair storage databases storage data as a hash table where each key is unique, and the value can be JSON, BLOB (Binary Large Objects) string, etc. For example, a key-value pair may contain a key like “website” associated with a value like “Guru99”.



It is one of the most basic NoSQL database examples. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. Redis, Dynamo, Riak are some NoSQL examples of key-value store DataBases. They are all based on Amazon’s Dynamo paper.

**COLUMN-BASED**

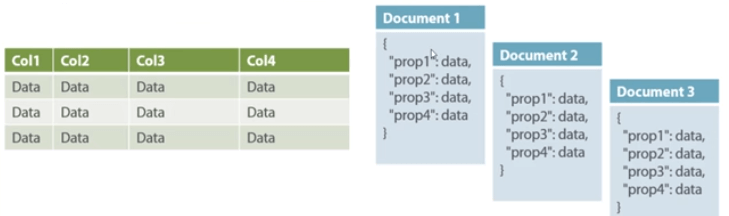
Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously. They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.



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

**DOCUMENT-ORIENTED**

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

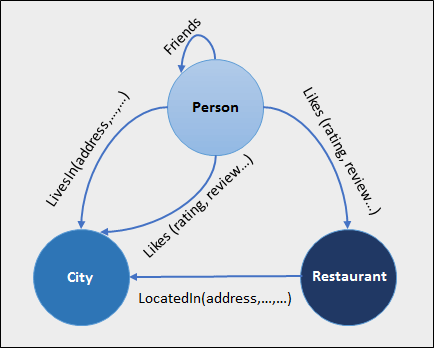


In this diagram on the left, we can see a SQL database, and at right a document -oriented NoSQL. The document type is mostly used for CMS systems, blogging platforms, real-time analytics and e-commerce applications. It should not be used for complex transactions which require multiple operations or queries against varying aggregate structures.

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

**GRAPH-BASED**

A graph-based database stores entity as well the relationships amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge have a unique identifier.



Compared to a relational database where tables are loosely connected, a graph database is multi-relational in nature. Traversing relationships is fast as they are already captured into the DB, and there is no need to calculate them. Graph databases are mostly used for social networks, logistics and spatial data.

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

**ADVANTAGES OF NOSQL**

* Can be used as Primary or Analytic Data source.
* BigData capability.
* No single point of failure.
* Easy replication.
* No need for separate caching layer.
* Fast performance and horizontal scalability.
* Can handle structured, semi-structured, and unstructured data with equal effect.
* Flexible and easy to used Object-oriented programming.
* Supports key developer languages and platforms.
* Simple implementation.
* It can serve as the primary data source for online applications.
* Flexible schema design which can easily be altered without downtime or service disruption.

**DISADVANTAGES OF NOSQL**

* No standardization rules.
* Limited query capabilities.
* Prone to lack of consistency.
* The RDBMS tools are comparatively mature and the learning curve is stiff for new developers.

**WHEN SHOULD NOSQL BE USED:**

* When a huge amount of data needs to be stored and retrieved.
* The relationship between the data you store is not that important.
* The data changes over time and is not structured.
* The data changes over time and is not structured.
* Support of Constraints and Joins is not required at the database level.
* The data is growing continuously and you need to scale the database regularly to handle the data.

In conclusion, NoSQL databases offer several benefits over traditional relational databases, such as scalability, flexibility and cost-effectiveness. However, they also have several drawbacks, such as lack of standardization, lack of ACID compliance, and lack of support for complex queries. Then choosing a database for a specific application, it is important to weigh the benefits and drawbacks carefully to determine the best fit.

