UNIT - III

Big Data Anayltics :

Big data analytics is the process of examining large data sets to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. The analytical findings can lead to more effective marketing, new revenue opportunities, better customer service, improved operational efficiency, competitive advantages over rival organizations and other business benefits.

The primary goal of big data analytics is to help companies make more informed business decisions by enabling data scientists, predictive modelers and other analytics professionals to analyze large volumes of transaction data, as well as other forms of data that may be untapped by conventional business intelligence (BI) programs. That could include Web server logs and Internet clickstream data, social media content and social network activity reports, text from customer emails and survey responses, mobile-phone call detail records and machine data captured by sensors connected to the Internet of Things.

Semi-structured and unstructured data may not fit well in traditional data warehouses based on relational databases. Furthermore, data warehouses may not be able to handle the processing demands posed by sets of big data that need to be updated frequently or even continually -- for example, real-time data on the performance of mobile applications or of oil and gas pipelines. As a result, many organizations looking to collect, process and analyze big data have turned to a newer class of technologies that includes Hadoop and related tools such as YARN, MapReduce, Spark, Hive and Pig as well as NoSQL databases. Those technologies form the core of an open source software framework that supports the processing of large and diverse data sets across clustered systems.

Analytics 1.0

Velleity is a word that has dropped out of the general vocabulary, unfortunately. I am attempting to bring it back, as it has more relevance now than ever.

Velleity is a desire to see something done, but not enough desire to make it happen. Wow!

Reporting v Analyzing
In my analytics training classes I ask the attendees about their analytics

reporting routine. Sometimes I make fun of the in-house analyst that has to report numbers like unique visitors, hits and sessions. Then, they spend the rest of the month justifying why that number was higher or lower than the prior month. Unfortunately, as I make that statement I see too many heads nodding in agreement. I feel for those people who are locked in a never ending cycle of velleity. Companies that understand that analytics can be valuable, but not enough to change their culture, provide the analyst with the tools they need, or provide them with the freedom to make changes that will improve the profits of the company itself.

These are what I call “Caveman Analytics.”

Unfortunately, too many businesses are trapped into thinking that big numbers are impressive. Big numbers = big business, right? But what do you do with those numbers? How does that affect your strategy?

Questioning the Strategy
As soon as someone starts asking questions, the house of straw blows away. Simply reporting numbers is not an analytics strategy, and it certainly will not lead to any amount of website improvement. And yes, I do know of some companies that include “Hits” in their monthly reporting.
(If that is you, reporting hits, stop. Hits are not a count of any relevance for your marketing. It’s just a big number.)

Questioning As a Strategy
Questions are the foundation for our learning. They expose motivations and require explanations. Many corporations and businesses are famous for encouraging the heads-down, lock-step agreement survival tactics. Asking questions is not popular, nor is it encouraged. Asking questions is perceived as rebellion, rather than progress. In the 60’s the radicals told us to question authority. Now that they are in authority, the last thing they want is to be questioned.

Neil Postman speaks about the importance of questioning, saying “question-asking is the most significant tool human beings have.” Even more important that software, servers and summaries, questions are an analysts’ primary tool. Postman even suggested that the reason why we don’t teach the ability to question in schools, is because eventually the students will question the teacher. Questions are subversive, but they result in people finding answers.

Questions – The Cure for Velleity
Velleity is what keeps companies locked in this mindset of reporting useless numbers. Desiring, even expecting to someday have an epiphany of change, but not willing to change the mindset or the culture of locked-in reporting to

achieve it. Nor are they willing to ask the hard questions in order to uncover what must be done.

Analytics 2.0 :

Analytics 2.0 or Big Data Analytics in 2000s with competitive insight which added complex queries along with forward-looking and predictive views leveraging both structured and unstructured data such as social media, mobile data, call center logs. Big data that couldn't fit or be analyzed fast enough on a centralized platform was processed with Hadoop, an open source software framework for fast batch data processing across parallel servers either in cloud or on premise. To deal with unstructured data, companies also turned to a new class of databases known as NoSQL, or not only SQL to support key/value, document, graph, columnar, and

geospatial data. Other big data technologies introduced during this period include "in memory" analytics for fast analysis where the data is managed and processed in memory rather than on disk.

Today, massive amounts of data are being created at the edge of the network and the traditional ways of doing analytics is no longer viable. Companies that make things, move things, consume things, or work with customers, they have increasing amounts of data on those devices and activities. Every device, shipment, and consumer leaves a trail referred to as data exhaust. It's just not feasible to keep moving very large. .amounts of raw data to centralized data stores – it is too big, changing too fast, and hyper distributed. And it is ushering in what's referred to as Analytics 3.0

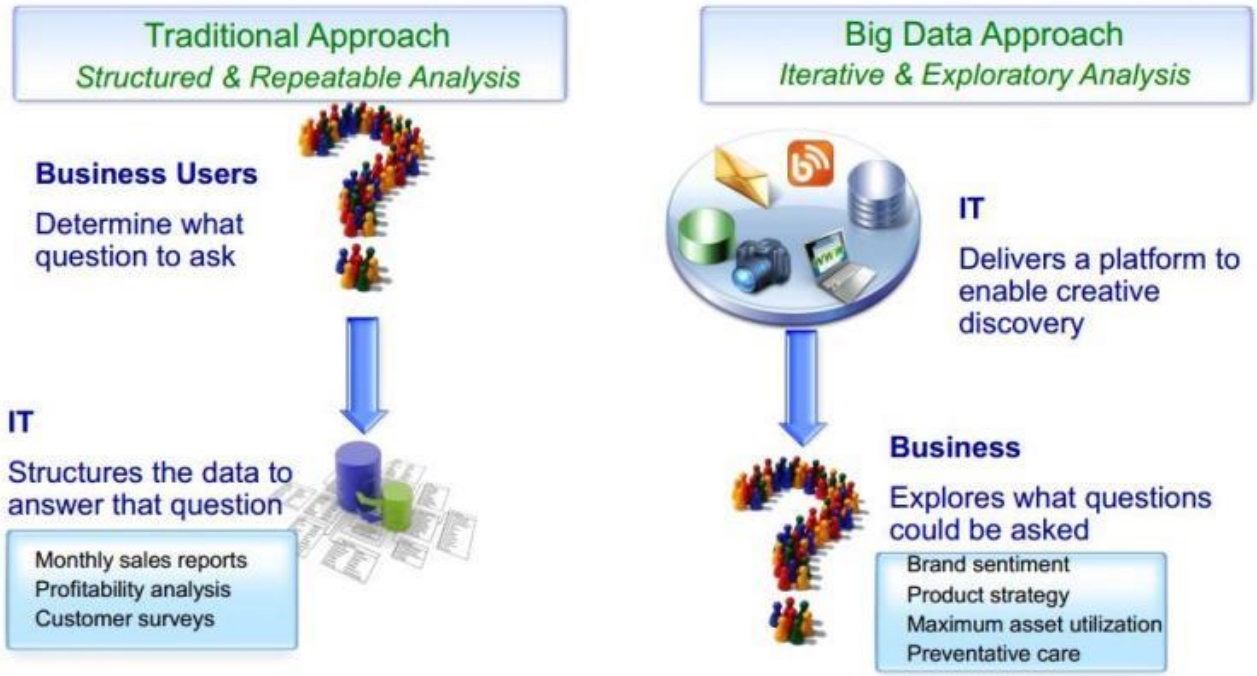
Analytics 3.0:

Analytics 3.0, which is about connecting data generated at the edge with data that is stored in enterprise data centers. Consequently, some aspects of both storage and analytical capabilities have to be closer to the edge. Analytics 3.0 is essentially a combination of traditional business intelligence, big data and Internet of Things (IoT) distributed throughout the network. Companies have the ability to analyze those data sets for the benefit of customers and monetize on them. They also have the ability to embed advanced analytics and optimization in near real-time into every business decision made at the front lines of their operations.

What is the next era for Analytics? There is already hype about Quantum Analytics, an Analytics based on quantum computing. In theory, quantum computing takes advantage of subatomic and molecule interactions to perform operations on data to solve certain problems far faster than a binary conventional computers could. IBM, Google, Microsoft have built research labs

and these companies and startups together have made some progress on the materials, designs and methods needed to make quantum computing possible. It will be years before a full-scale quantum analytics takes off. Frankly, companies are just ramping up their adoption of Big Data (Analytics 2.0) and IoT Analytics (Analytics 3.0), so we will just have to wait for a while for the thrill of Quantum Analytics to enter mainstream.

Traditional BI vs. Big DataEnvirnoment:



I know that not everyone will agree with my definition of Business Intelligence, but my objective is to simplify things; there is enough confusion out there. Besides, who is the authority on a terminology that its traditional frame of

reference is outdated and doesn't cover the entire spectrum of the value that intelligent-data can bring to businesses today?

Business Intelligence (BI) encompasses a variety of tools and methods that can help organizations make better decisions by analyzing “their” data. Therefore, Data Analytics falls under BI. Big Data, if used for the purpose of Analytics falls under BI as well.

Let’s say I work for the Center for Disease Control and my job is to analyze the data gathered from around the country to improve our response time during flu season. Suppose we want to know about the geographical spread of flu for the last winter (2012). We run some BI reports and it tells us that the state of New York had the most outbreaks. Knowing that information we might want to better prepare the state for the next winter. Theses types of queries examine past events, are most widely used, and fall under the Descriptive Analytics category.

Now, we just purchased an interactive visualization tool and I am looking at the map of the United States depicting the concentration of flu in different states for the last winter. I click on a button to display the vaccine distribution. There it is; I visually detected a direct correlation between the intensity of flu outbreak with the late shipment of vaccines. I noticed that the shipments of vaccine for the state of New York were delayed last year. This gives me a clue to further investigate the case to determine if the correlation is causal. This type of analysis falls under Diagnostic Analytics (discovery).

We go to the next phase which is Predictive Analytics. PA is what most people in the industry refer to as Data Analytics. It gives us the probability of different outcomes and it is future-oriented. The US banks have been using it for things like fraud detection. The process of distilling intelligence is more complex and it requires techniques like Statistical Modeling. Back to our examples, I hire a Data Scientist to help me create a model and apply the data to the model in order to identify causal relationships and correlations as they relate to the spread of flu for the winter of 2013. Note that we are now taking about the future. I can use my visualization tool to play around with some variables such as demand, vaccine production rate, quantity… to weight the pluses and minuses of different decisions insofar as how to prepare and tackle the potential problems in the coming months.

The last phase is the Prescriptive Analytics and that is to integrate our tried-and-true predictive models into our repeatable processes to yield desired outcomes. An automated risk reduction system based on real-time data received from the sensors in a factory would be a good example of its use case.

Finally, here is an example of Big Data. Suppose it’s December 2013 and it happens to be a bad year for the flu epidemic. A new strain of the virus is

wreaking havoc, and a drug company has produced a vaccine that is effective in combating the virus. But, the problem is that the company can't produce them fast enough to meet the demand. Therefore, the Government has to prioritize its shipments. Currently the Government has to wait a considerable amount of time to gather the data from around the country, analyze it, and take action. The process is slow and inefficient. The following includes the contributing factors. Not having fast enough computer systems capable of gathering and storing the data (velocity), not having computer systems that can accommodate the volume of the data pouring in from all of the medical centers in the country (volume), and not having computer systems that can process images, i.e, x-rays (variety).

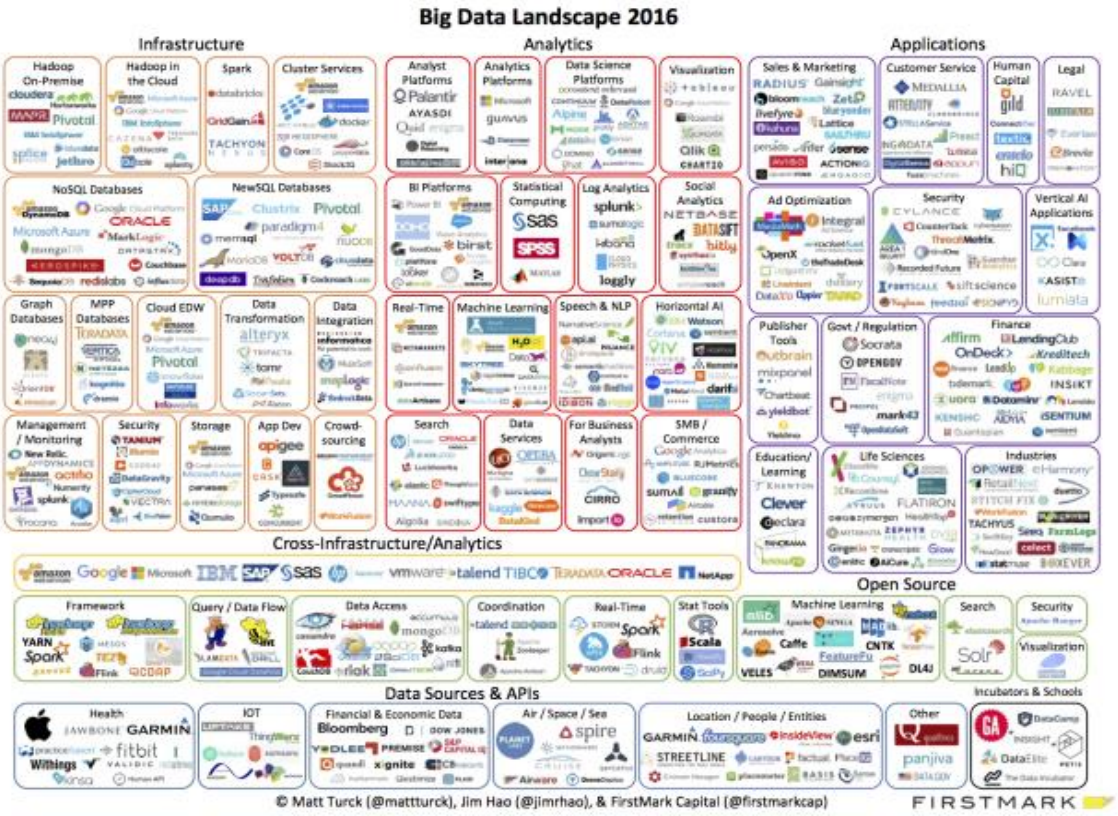
Big Data technology changed all of that. It solved the velocity-volume-variety problem. We now have computer systems that can handle "Big Data". The Center for Disease Control may receive the data from hospitals and doctor offices in real-time and Data Analytics Software that sits on the top of Big Data computer system could generate actionable items that can give the Government the agility it needs in times of crises.

Big Data technology Landscape:

2015 was a year of significant change in the "Art of the Possible" with Analytics. Norms about Analytics are evolving, and as they do, leading to wholesale business model transformation and cultural change at some workplaces. This change is driven not only by fast-moving technology, but also by new techniques to get value from data.

2016 is turning into the year of "democratization of data", AI and Deep Learning. The state of the art is around Deep learning (also known as structured learning, hierarchical learning or machine learning), which are essentially efficient algorithms for unsupervised or semi-supervised feature learning and hierarchical feature extraction.
Customer intelligence is the hot space. Commoditization pressures and shifting consumer expectations have inspired many companies to creatively use data and analytics to enrich their core products and services, a phenomenon called "wrapping". Companies wrap offerings with information to differentiate them and to add value to customers. The best companies build distinctive competencies for wrapping.

However, there's a lot of technology in the world, my friend. The 2016 edition of Matt Turck& Jim Hao Big Data landscape supergraphic is shown below.



NoSQL Databases :

What does NoSQL mean and how do you categorize these databases? NoSQL means Not Only SQL, implying that when designing a software solution or product, there are more than one storage mechanism that could be used based on the needs. NoSQL was a hashtag (#nosql) choosen for a meetup to discuss these new databases. The most important result of the rise of NoSQL is Polyglot Persistence. NoSQL does not have a prescriptive definition but we can make a set of common observations, such as:

- Not using the relational model
- Running well on clusters
- Mostly open-source
- Built for the 21st century web estates
- Schema-less

Why NoSQL Databases

Application developers have been frustrated with the impedance mismatch between the relational data structures and the in-memory data structures of the application. Using NoSQL databases allows developers to develop without having to convert in-memory structures to relational structures.

There is also movement away from using databases as integration points in favor of encapsulating databases with applications and integrating using services.

The rise of the web as a platform also created a vital factor change in data storage as the need to support large volumes of data by running on clusters.

Relational databases were not designed to run efficiently on clusters.

The data storage needs of an ERP application are lot more different than the data storage needs of a Facebook or an Etsy, for example.

Aggregate Data Models:
Relational database modelling is vastly different than the types of data structures that application developers use. Using the data structures as modelled by the developers to solve different problem domains has given rise to movement away from relational modelling and towards aggregate models, most of this is driven by *Domain Driven Design*, a book by Eric Evans. An aggregate is a collection of data that we interact with as a unit. These units of data or aggregates form the boundaries for ACID operations with the database, Key-value, Document, and Column-family databases can all be seen as forms of aggregate-oriented database.
Aggregates make it easier for the database to manage data storage over clusters, since the unit of data now could reside on any machine and when retrieved from the database gets all the related data along with it. Aggregate-oriented databases work best when most data interaction is done with the same aggregate, for example when there is need to get an order and all its details, it better to store order as an aggregate object but dealing with these aggregates to get item details on all the orders is not elegant.

Aggregate-oriented databases make inter-aggregate relationships more difficult to handle than intra-aggregate relationships. Aggregate-ignorant databases are better when interactions use data organized in many different formations.
Aggregate-oriented databases often compute materialized views to provide data organized differently from their primary aggregates. This is often done with map-reduce computations, such as a map-reduce job to get items sold per day.

Distribution Models:
Aggregate oriented databases make distribution of data easier, since the distribution mechanism has to move the aggregate and not have to worry about related data, as all the related data is contained in the aggregate. There are two styles of distributing data:

- Sharding: Sharding distributes different data across multiple servers, so each server acts as the single source for a subset of data.
- Replication: Replication copies data across multiple servers, so each bit of data can be found in multiple places. Replication comes in two forms,
  - Master-slave replication makes one node the authoritative copy that handles writes while slaves synchronize with the master and may handle reads.
  - Peer-to-peer replication allows writes to any node; the nodes coordinate to synchronize their copies of the data.

Master-slave replication reduces the chance of update conflicts but peer-to-peer replication avoids loading all writes onto a single server creating a single point of failure. A system may use either or both techniques. Like Riak database shards the data and also replicates it based on the replication factor.

CAP theorem:
In a distributed system, managing consistency(C), availability(A) and partition toleration(P) is important, Eric Brewer put forth the CAP theorem which states that in any distributed system we can choose only two of consistency, availability or partition tolerance. Many NoSQL databases try to provide options where the developer has choices where they can tune the database as per their needs. For example if you consider Riak a distributed key-value database. There are essentially three variables r, w, n where

- r=number of nodes that should respond to a read request before its considered successful.
- w=number of nodes that should respond to a write request before its considered successful.
- n=number of nodes where the data is replicated aka replication factor.

In a Riak cluster with 5 nodes, we can tweak the r,w,n values to make the system very consistent by setting r=5 and w=5 but now we have made the cluster susceptible to network partitions since any write will not be considered successful when any node is not responding. We can make the same cluster highly available for writes or reads by setting r=1 and w=1 but now consistency can be compromised since some nodes may not have the latest copy of the data. The CAP theorem states that if you get a network partition, you have to trade off availability of data versus consistency of data. Durability can also be traded off against latency, particularly if you want to survive failures with replicated data.

NoSQL databases provide developers lot of options to choose from and fine tune the system to their specific requirements. Understanding the requirements of how the data is going to be consumed by the system, questions such as is it read heavy vs write heavy, is there a need to query data with random query parameters, will the system be able handle inconsistent data.

Understanding these requirements becomes much more important, for long we have been used to the default of RDBMS which comes with a standard set of features no matter which product is chosen and there is no possibility of choosing some features over other. The availability of choice in NoSQL databases, is both good and bad at the same time. Good because now we have choice to design the system according to the requirements. Bad because now you have a choice and we have to make a good choice based on requirements and there is a chance where the same database product may be used properly or not used properly.

An example of feature provided by default in RDBMS is transactions, our development methods are so used to this feature that we have stopped thinking about what would happen when the database does not provide transactions. Most NoSQL databases do not provide transaction support by default, which means the developers have to think how to implement transactions, does every write have to have the safety of transactions or can the write be segregated into "critical that they succeed" and "its okay if I lose this write" categories. Sometimes deploying external transaction managers like ZooKeeper can also be a possibility.
Types of NoSQL Databases:
NoSQL databases can broadly be categorized in four types.

Key-Value databases

Key-value stores are the simplest NoSQL data stores to use from an API perspective. The client can either get the value for the key, put a value for a key, or delete a key from the data store. The value is a blob that the data store just stores, without caring or knowing what's inside; it's the responsibility of the application to understand what was stored. Since key-value stores always use primary-key access, they generally have great performance and can be easily scaled.

Some of the popular key-value databases are Riak, Redis (often referred to as Data Structure server), Memcached and its flavors, Berkeley DB, upscaledb (especially suited for embedded use), Amazon DynamoDB (not open-source), Project Voldemort and Couchbase.
All key-value databases are not the same, there are major differences between these products, for example: Memcached data is not persistent while in Riak it is, these features are important when implementing certain solutions. Lets consider we need to implement caching of user preferences, implementing them in memcached means when the node goes down all the data is lost and needs to be refreshed from source system, if we store the same data in Riak we may not need to worry about losing data but we must also consider how to update stale data. Its important to not only choose a key-value database based on your requirements, it's also important to choose which key-value database.

NoSQL Vs. RDBMS :

When a disruptive technology enters into the market and gains widespread adoption, it is typically because it offers a better, more efficient and cheaper solution. NoSQL databases disrupted the database market by offering a more flexible, scalable, and less expensive alternative to relational databases. They also were built to better handle the requirements of Big Data applications.

NoSQL databases differ from older, relational technology in four main areas:

- Data models: A NoSQL database lets you build an application without having to define the schema first unlike relational databases which make you define your schema before you can add any data to the system. No predefined schema makes NoSQL databases much easier to update as your data and requirements change.
- Data structure: Relational databases were built in an era where data was fairly structured and clearly defined by their relationships. NoSQL databases are designed to handle unstructured data (e.g., texts, social media posts, video, email) which makes up much of the data that exists today.
- Scaling: It's much cheaper to scale a NoSQL database than a relational database because you can add capacity by scaling out over cheap, commodity servers. Relational databases, on the other hand, require a single server to host your entire database. To scale, you need to buy a bigger, more expensive server.
- Development model: NoSQL databases are open source whereas relational databases typically are closed source with licensing fees baked into the use of their software. With NoSQL, you can get started on a project without any heavy investments in software fees upfront.

It's no surprise then that NoSQL databases are so popular these days. And when companies look to a NoSQL database for their mission-critical applications, they choose MongoDB. MongoDB is the most downloaded NoSQL database with 10 million downloads and hundreds of thousands of deployments. MongoDB is also the only database of this new generation that combines the innovations of NoSQL with the best aspects of relational databases in its Nexus Architecture.

NewSQL :

NEWSQL
The term NewSQL is not quite as broad as NoSQL. NewSQL systems all start with the relational data model and the SQL query language, and they all try to address some of the same sorts of scalability, inflexibility or lack-of-focus that has driven the NoSQL movement. Many offer stronger consistency guarantees.

But within this group there are many differences. HANA was created to be a business reporting powerhouse that could also handle a modest transactional workload, a perfect fit for SAP deployments. Hekaton adds sophisticated in-memory processing to the more traditional Microsoft SQL Server. Both systems are non-clustering for now, and both are designed to replace or enhance OldSQL deployments directly.

NuoDB set out to be a cluster-first SQL database with a focus on cloud-ops: run on many nodes across many datacenters and let the underlying system manage data locality and consistency for you. This comes at a cost in performance and consistency for arbitrary workloads. For workloads that are closer to key-value, global data management is a more tractable problem. NuoDB is the closest to being called eventually consistent of the NewSQL systems.

Other systems focus on clustered analytics, such as MemSQL. Distributed, with MySQL compatibility, MemSQL often offers faster OLAP analytics than all-in-one OldSQL systems, with higher concurrency and the ability to update data as it's being analyzed.

VoltDB, the most mature of these systems, combines streaming analytics, strong ACID guarantees and native clustering. This allows VoltDB to be the system-of-record for data-intensive applications, while offering an integrated high-throughput, low-latency ingestion engine. It's a great choice for policy enforcement, fraud/anomaly detection, or other fast-decisioning apps.

THE NEED FOR SPEED: FAST IN-MEMORY SQL

Perhaps you have gigabytes to terabytes of data that needs high-speed transactional access. You have an incoming event stream (think sensors, mobile phones, network access points) and need per-event transactions to compute responses and analytics in real time. Your problem follows a pattern of "ingest, analyze, decide," where the analytics and the decisions must be calculated per-request and not post-hoc in batch processing. NewSQL systems that offer the scale of NoSQL with stronger consistency may be the right choice.

THE NEWSQL ADVANTAGES:

- Minimize application complexity stronger consistency and often full transactional support.
- Familiar SQL and standard tooling.
- Richer analytics leveraging SQL and extensions.
- Many systems offer NoSQL-style clustering with more traditional data and query models.

THE NEWSQL DISADVANTAGES:

- No NewSQL systems are as general-purpose as traditional SQL systems set out to be.
- In-memory architectures may be inappropriate for volumes exceeding a few terabytes.
- Offers only partial access to the rich tooling of traditional SQL systems.

SUMMING IT UP

As a general rule of thumb, consider evaluating NoSQL offerings if you favor availability or have special data model needs. Consider NewSQL if you'd like the at-scale speed of NoSQL, but with stronger consistency and the familiar and powerful SQL query language.

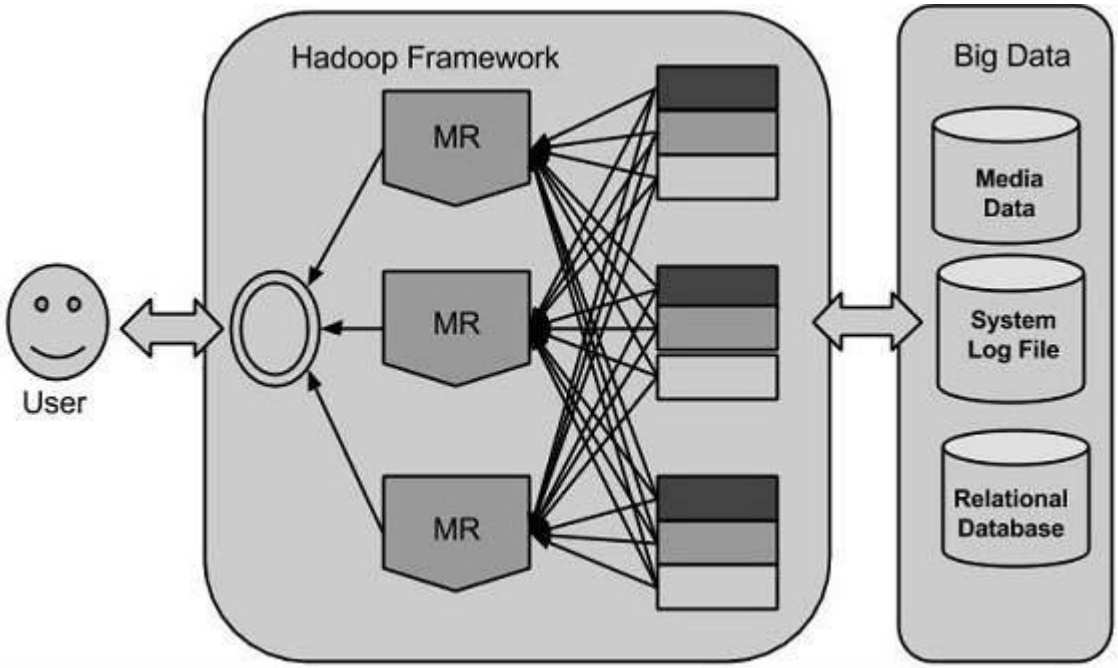
The froth in the data management space is substantial – and our tendency to talk in terms of categories (SQL, NoSQL, NewSQL) vs. problems makes it hard for software developers to understand what's in the toolbox. The current offerings of new databases are not all alike – and recognizing how the DNA behind each helps or hinders problem solvers is the key to choosing the best solution.

Hadoop :

Doug Cutting, Mike Cafarella and team took the solution provided by Google and started an Open Source Project called HADOOP in 2005 and Doug named it after his son's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.

Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capable enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge

amounts of data.



Hadoop 1.0 vs. Hadoop 2.0 :

Hadoop – the solution for deciphering the avalanche of Big Data – has come a long way from the time Google published its paper on Google File System in 2003 and MapReduce in 2004. It created waves with its scale-out and not scale-up strategy. Inroads from Doug Cutting and team at Yahoo and Apache Hadoop project resulted in popularizing MapReduce programming – which is intensive in I/O and is constrained in interactive analysis and graphics support. This paved the way for further evolving of Hadoop 1 to Hadoop 2. The following table describes the major differences between them:

1 No	Hadoop 1	Hadoop 2
1	Supports MapReduce (MR) processing model only. Does not support non MR tools	Allows to work in MR as well as other distributed computing models like Spark, Hama, Giraph, Message Passing Interface) MPI &HBase coprocessors.
2	MR does both processing and cluster-resource management.	YARN (Yet Another Resource Negotiator) does cluster resource management and processing is done using different processing

		models.
3	Has limited scaling of nodes. Limited to 4000 nodes per cluster	Has better scalability. Scalable up to 10000 nodes per cluster
4	Works on concepts of slots – slots can run either a Map task or a Reduce task only.	Works on concepts of containers. Using containers can run generic tasks.
5	A single Namenode to manage the entire namespace.	Multiple Namenode servers manage multiple namespace.
6	Has Single-Point-of-Failure (SPOF) – because of single Namenode- and in case of Namenode failure, needs manual intervention to overcome.	Has feature to overcome SPOF with a standby Namenode and in case of Namenode failure, it is configured for automatic recovery.
7	MR API is compatible with Hadoop 1x. A program written in Hadoop1 executes in Hadoop1x without any additional files.	MR API requires additional files for a program written in Hadoop1x to execute in Hadoop2x.
8	Has a limitation to serve as a platform for event processing, streaming and real time operations.	Can serve as a platform for a wide variety of data analytics-possible to run event processing, streaming and real time operations.
9	A Namenode failure affects the stack.	The Hadoop stack – Hive, Pig, HBase etc. are all equipped to handle Namenode failure.
10	Does not support Microsoft Windows	Added support for Microsoft windows

Hadoop 1

1. Hadoop 1.x Supports only MapReduce (MR) processing model.it Does not support non-MR tools.
2. MR does both processing and cluster resource management.
3. 1.x Has limited scaling of nodes. Limited to 4000 nodes per cluster.
4. Works on concepts of slots – slots can run either a Map task or a Reduce task only.

5. A single Namenode to manage the entire namespace.
6. 1.x Has Single-Point-of-Failure (SPOF) – because of single Namenode- and in case of Namenode failure, needs manual intervention to overcome.
7. MR API is compatible with Hadoop 1x. A program written in Hadoop1 executes in Hadoop1x without any additional files.
8. 1.x Has a limitation to serve as a platform for event processing, streaming and real-time operations.

Hadoop 2

1. Hadoop 2.x Allows to work in MR as well as other distributed computing models like Spark, Hama, Giraph, Message Passing Interface) MPI &HBase coprocessors.
2. YARN (Yet Another Resource Negotiator) does cluster resource management and processing is done using different processing models.
3. 2.x Has better scalability. Scalable up to 10000 nodes per cluster.
4. Works on concepts of containers. Using containers can run generic tasks.
5. Multiple Namenode servers manage multiple namespace.
6. 2.x Has feature to overcome SPOF with a standby Namenode and in case of Namenode failure, it is configured for automatic recovery.
7. MR API requires additional files for a program written in Hadoop1x to execute in Hadoop2x.
8. Can serve as a platform for a wide variety of data analytics-possible to run event processing, streaming and real time operations.

Exercises, Data Science ismultidisciplinary, Data Scientist - Your new best friend :

Data science – discovery of data insight

This aspect of data science is all about uncovering findings from data. Diving in at a granular level to mine and understand complex behaviors, trends, and inferences. It's about surfacing hidden insight that can help enable companies to make smarter business decisions. For example:

- Netflix data mines movie viewing patterns to understand what drives user interest, and uses that to make decisions on which Netflix original series to produce.
- Target identifies what are major customer segments within it's base and the unique shopping behaviors within those segments, which helps to guide messaging to different market audiences.
- Proctor & Gamble utilizes time series models to more clearly understand future demand, which help plan for production levels more optimally.

How do data scientists mine out insights? It starts with data exploration. When given a challenging question, data scientists become detectives. They investigate leads and try to understand pattern or characteristics within the data. This requires a big dose of analytical creativity.

Then as needed, data scientists may apply quantitative technique in order to get a level deeper – e.g. inferential models, segmentation analysis, time series forecasting, synthetic control experiments, etc. The intent is to scientifically piece together a forensic view of what the data is really saying.

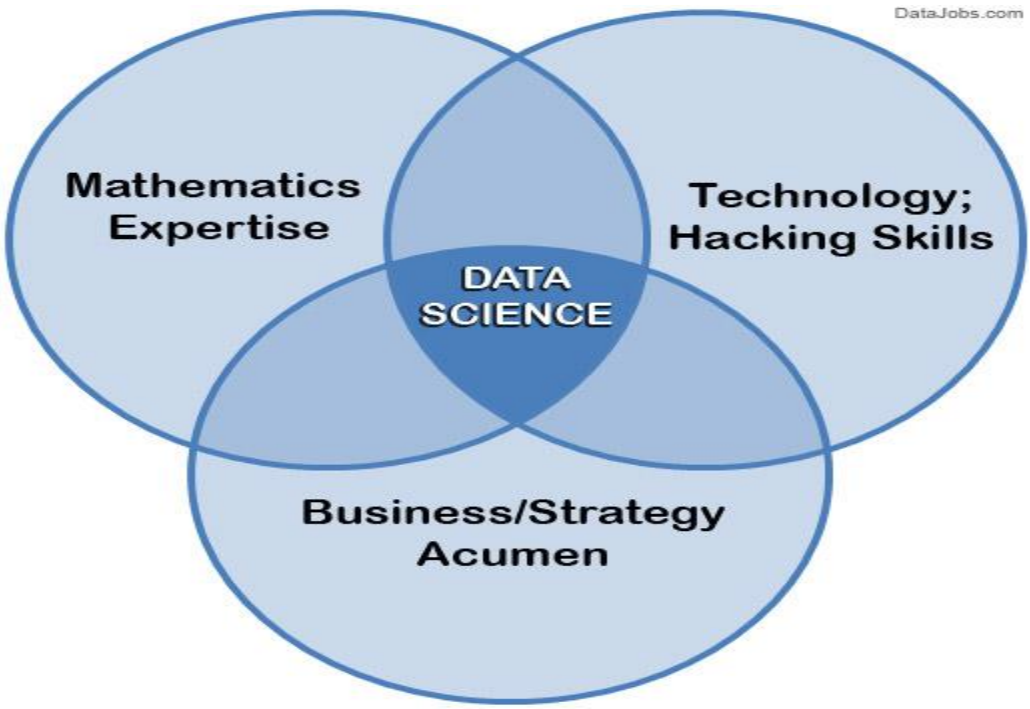
This data-driven insight is central to providing strategic guidance. In this sense, data scientists act as consultants, guiding business stakeholders on how to act on findings.

Data science – development of data product
A "data product" is a technical asset that: (1) utilizes data as input, and (2) processes that data to return algorithmically-generated results. The classic example of a data product is a recommendation engine, which ingests user data, and makes personalized recommendations based on that data. Here are some examples of data products:

- Amazon's recommendation engines suggest items for you to buy, determined by their algorithms. Netflix recommends movies to you. Spotify recommends music to you.
- Gmail's spam filter is data product – an algorithm behind the scenes processes incoming mail and determines if a message is junk or not.
- Computer vision used for self-driving cars is also data product – machine learning algorithms are able to recognize traffic lights, other cars on the road, pedestrians, etc.

This is different from the "data insights" section above, where the outcome to that is to perhaps provide advice to an executive to make a smarter business decision. In contrast, a data product is technical functionality that encapsulates an algorithm, and is designed to integrate directly into core applications. Respective examples of applications that incorporate data product behind the scenes: Amazon's homepage, Gmail's inbox, and autonomous driving software.

Data scientists play a central role in developing data product. This involves building out algorithms, as well as testing, refinement, and technical deployment into production systems. In this sense, data scientists serve as technical developers, building assets that can be leveraged at wide scale.
What is data science – the requisite skill set
Data science is a blend of skills in three major areas:



Mathematics Expertise
At the heart of mining data insight and building data product is the ability to view the data through a quantitative lens. There are textures, dimensions, and correlations in data that can be expressed mathematically. Finding solutions utilizing data becomes a brain teaser of heuristics and quantitative technique. Solutions to many business problems involve building analytic models grounded in the hard math, where being able to understand the underlying mechanics of those models is key to success in building them.

Also, a misconception is that data science all about statistics. While statistics is important, it is not the only type of math utilized. First, there are two branches of statistics – classical statistics and Bayesian statistics. When most people refer to *stats* they are generally referring to *classical stats*, but knowledge of both types is helpful. Furthermore, many inferential techniques and machine learning algorithms lean on knowledge of linear algebra. For example, a popular method to discover hidden characteristics in a data set is SVD, which is grounded in matrix math and has much less to do with classical stats. Overall, it is helpful for data scientists to have breadth and depth in their knowledge of mathematics.

Technology and Hacking
First, let's clarify on that we are *not* talking about hacking as in breaking into computers. We're referring to the tech programmer subculture meaning of hacking – i.e., creativity and ingenuity in using technical skills to build things and find clever solutions to problems.

Why is hacking ability important? Because data scientists utilize *technology* in order to wrangle enormous data sets and work with complex algorithms, and it requires tools far more sophisticated than Excel. Data scientists need to be able to code — prototype quick solutions, as well as integrate with complex data systems. Core languages associated with data science include SQL, Python, R, and SAS. On the periphery are Java, Scala, Julia, and others. But it is not just knowing language fundamentals. A hacker is a technical ninja, able to creatively navigate their way through technical challenges in order to make their code work.

Along these lines, a data science hacker is a solid algorithmic thinker, having the ability to break down messy problems and recompose them in ways that are solvable. This is critical because data scientists operate within a lot of algorithmic complexity. They need to have a strong mental comprehension of high-dimensional data and tricky data control flows. Full clarity on how all the pieces come together to form a cohesive solution.

Strong Business Acumen

It is important for a data scientist to be a tactical business consultant. Working so closely with data, data scientists are positioned to learn from data in ways no one else can. That creates the responsibility to translate observations to shared knowledge, and contribute to strategy on how to solve core business problems. This means a core competency of data science is using data to cogently tell a story. No data-puking – rather, present a cohesive narrative of problem and solution, using data insights as supporting pillars, that lead to guidance.

Having this business acumen is just as important as having acumen for tech and algorithms. There needs to be clear alignment between data science projects and business goals. Ultimately, the value doesn't come from data, math, and tech itself. It comes from leveraging all of the above to build valuable capabilities and have strong business influence.