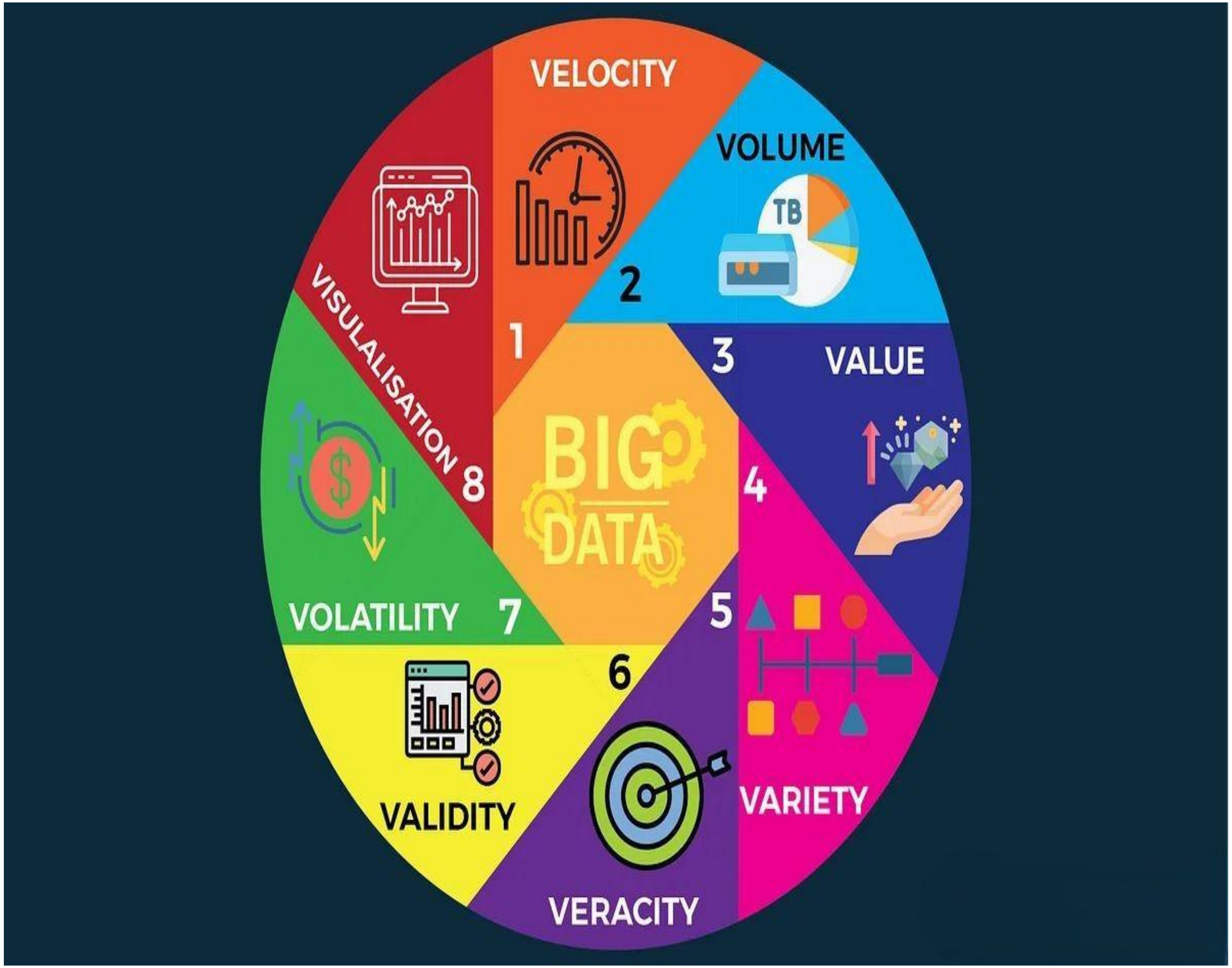


# CS 3032: Big Data

**Lec-6**

## In this Discussion . . .

- Exploring the Big Data Stack
  - Data Sources Layer
  - Ingestion Layer
  - Storage Layer

## Exploring the Big Data Stack

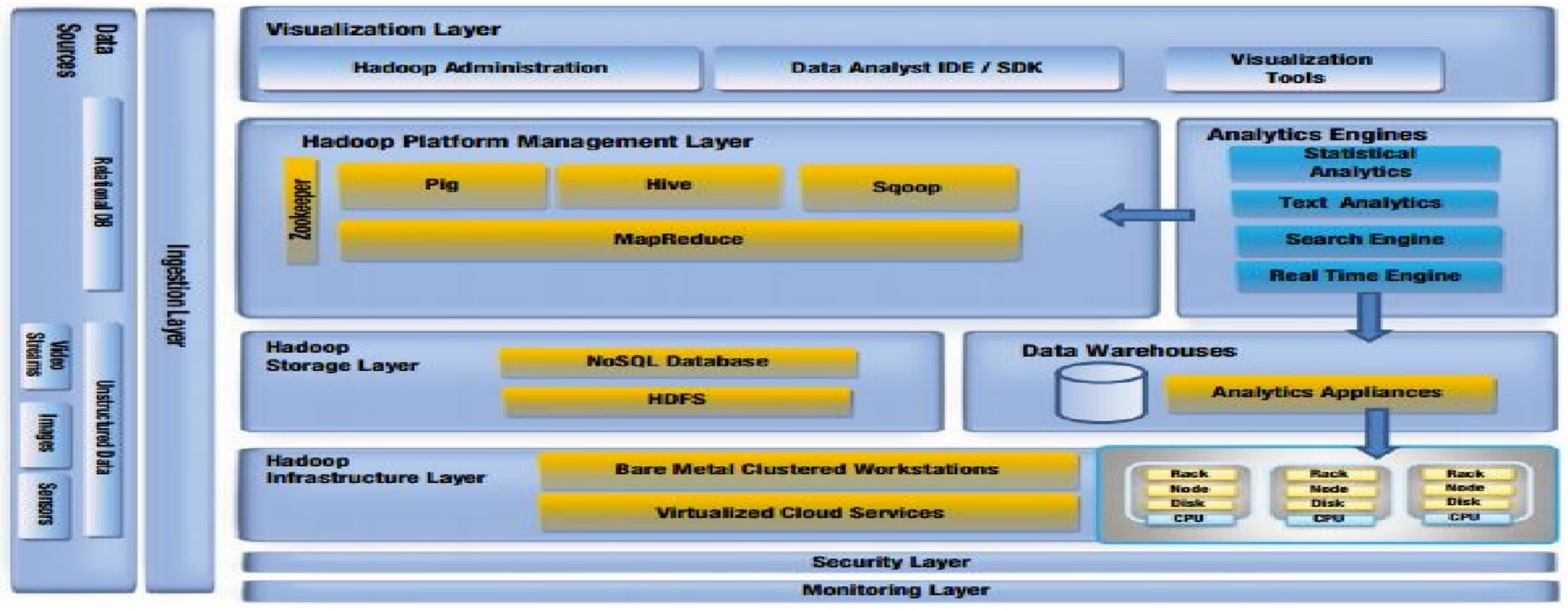
- The first step in the process of designing any data architecture is to create a model that should give a complete view of all required elements.

- Although, initially, creating a model may seem to be a time-consuming task, however, it can save a significant amount of time, effort, and rework  during the subsequent implementations.

## Exploring the Big Data Stack

- **Big Data analysis also needs the creation of a model or architecture, commonly known as the Big Data architecture.** Such architecture of the big data environment must fulfill all the foundational requirements and must be able to perform the following key functions:
  - Capturing data from different data sources
  - Cleaning and integrating data of different types & formats
  - Storing and organizing data
  - Analyzing data
  - Identifying relationship and patterns
  - Deriving conclusions based on the data analysis results

# Big Data Stack Layers



## Data Sources Layer

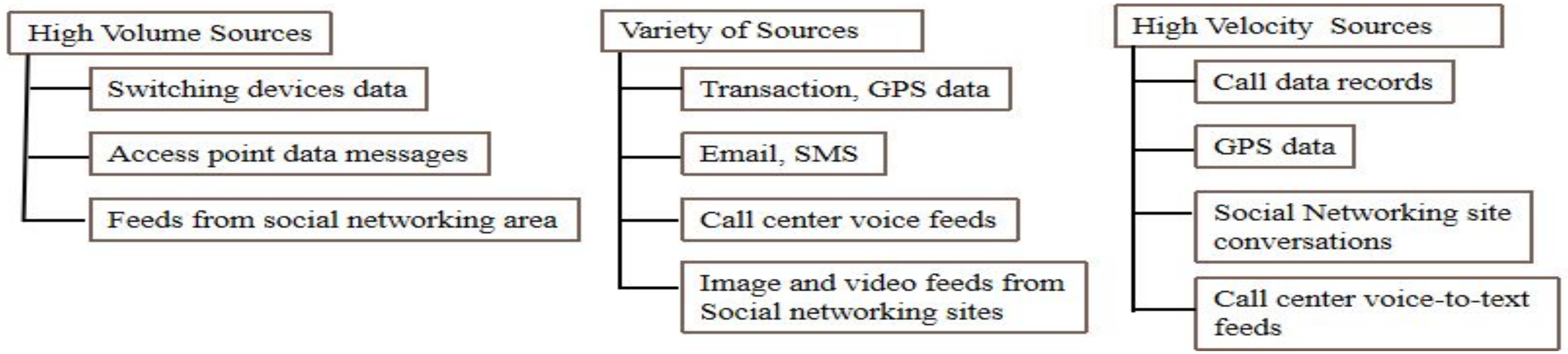
- **Data Sources layer** is responsible **for the absorption and integration of raw data incoming from the sources.**
- The velocity and variety of the incoming data can differ.
- This data needs to be validated and noise removal must be done before enterprises can use it in any logical sense.

## Data Sources Layer (Contd.)

- In other words, **the basic function of the data sources layer is to absorb and integrate the data from various sources, at varying velocity and in different formats.**
- It includes everything from sales records, customer database, feedback, social media channels, marketing list, email archives and any data obtained from monitoring or measuring aspects of operations.

## Data Sources Layer (Contd.)

- **Before this data is considered for big data stack, differentiation is required between the noise and relevant information.**
- Example of different data sources the telecom industry obtains its data:


High Volume Sources
Switching devices data
Access point data messages
Feeds from social networking area
Variety of Sources
Transaction, GPS data
Email, SMS
Call center voice feeds
Image and video feeds from Social networking sites
High Velocity Sources
Call data records
GPS data
Social Networking site conversations
Call center voice-to-text feeds

## Ingestion Layer

- **Data ingestion** is the **first step for building Data Pipeline and also the toughest task in the System of Big Data**.
- In this layer **we plan the way to ingest data flows from hundreds or thousands of sources into Data Center**. As the Data is coming from Multiple sources at variable speed, in different formats.
- That's why we should properly ingest the data for the successful business decisions making. It's rightly said that "If starting goes well, then, half of the work is already done."

## Ingestion Layer (Contd.)

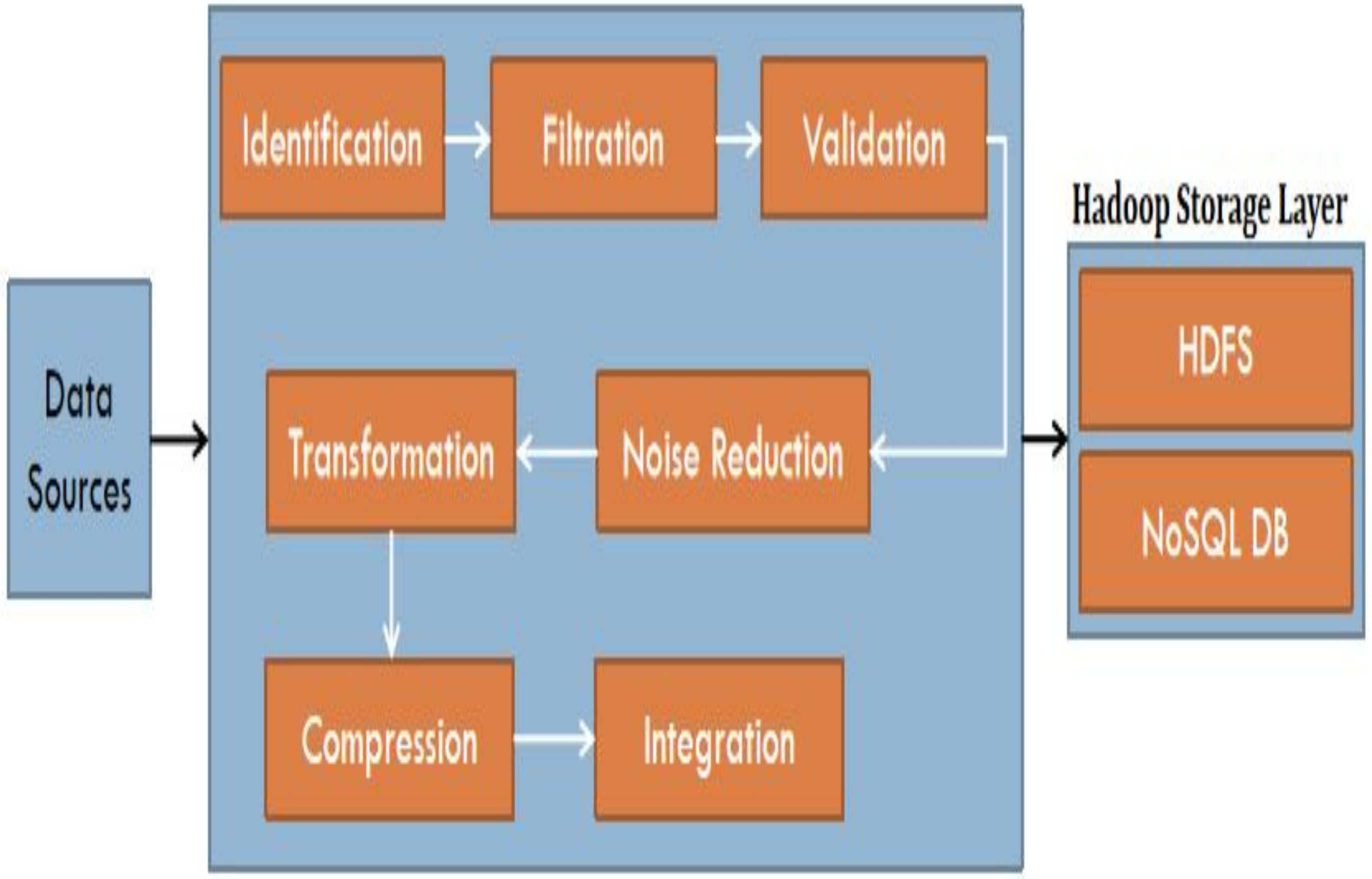
- Big Data Ingestion involves connecting to various data sources, extracting the data, and detecting the changed data.
- It’s about moving data — and especially the unstructured data — from where it is originated, into a system where it can be stored and analyzed.
- We can also say that Data Ingestion means taking data coming from multiple sources and putting it somewhere it can be accessed. It is the beginning of Data Pipeline where it obtains or import data for immediate use.

## Ingestion Layer (Contd.)

- Data can be streamed in real time or ingested in batches, When data is ingested in real time then, as soon as data arrives it is ingested immediately.
- When data is ingested in batches, data items are ingested in some chunks at a periodic interval of time.
- Ingestion is the process of bringing data into Data Processing system.
- Effective Data Ingestion process begins by prioritizing data sources, validating individual files and routing data items to the correct destination.

## Ingestion Layer (Contd.)

- In other words, this layer absorb the huge inflow of data and separates noise from relevant information.
- It handle huge volume, high velocity, and a variety of data.
- It validates, cleanses, transforms, reduces, and integrates the unstructured data into the Big Data stack for further processing.
- In this layer, the data passes through the following stages.



## Ingestion Layer (Contd.)

- In the ingestion layer, the data passes through the following stages:
    - **Identification:** At this stage, data is categorized into various known data formats or even unstructured data is assigned with default formats.
    - **Filtration:** At this stage, the information relevant for the enterprise is filtered on the basis of the Enterprise **Master Data Management** (MDM) **repository. MDM is a comprehensive method of enabling an enterprise to link all of its critical non-transactional data to a common point of reference.**

# Ingestion Layer (Contd.)

- In the ingestion layer, the data passes through the following stages:
    - **Validation –** At this stage, the filtered data is analyzed against MDM metadata.
    - **Noise Reduction–** At this stage, data is cleansed by removing the noise and minimizing the related disturbances.
    - **Transformation -** At this stage, data is split or combined on the basis of its type, contents, and the requirements of the organizations.

## Ingestion Layer (Contd.)

- In the ingestion layer, the data passes through the following stages:
  - **Compression** - At this stage, the size of the data is reduced without affecting its relevance for the required process and it guarantees of no adverse effects to the analysis result.
  - **Integration** - At this stage, the refined dataset is integrated with the Hadoop storage layer, which primarily consists of HDFS (Hadoop Distributed File System) and NoSQL databases.

## Storage Layer

- It is the place where Big Data lives.
- Hadoop is an open source framework used to store large volumes of data in a distributed manner across multiple machines.
- The **Hadoop storage layer** supports **fault-tolerance** and **parallelization**, **which enable high-speed distributed processing algorithms to execute over large-scale data.**

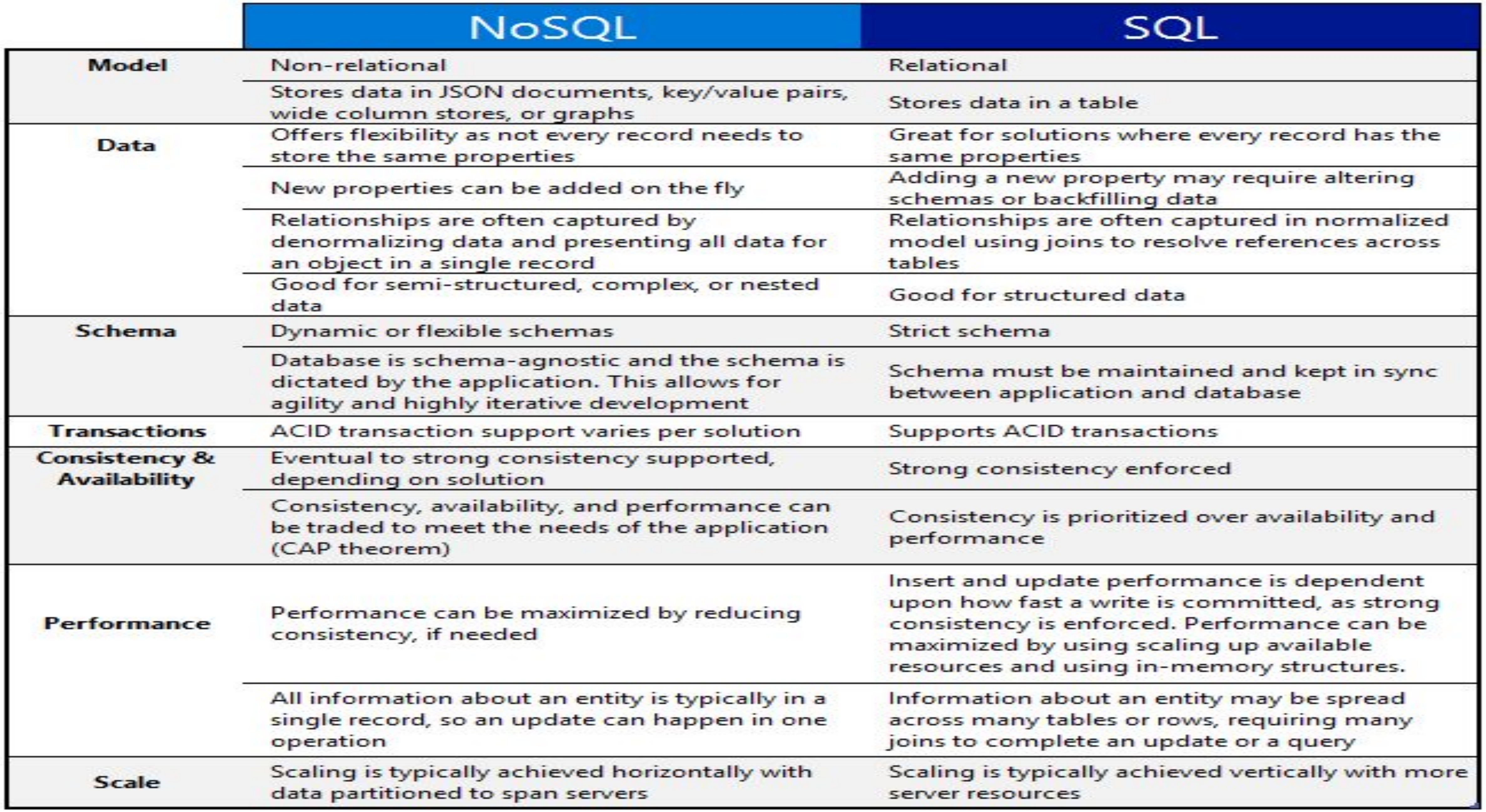
## Storage Layer (Contd.)

- Two major component of Hadoop:
  - **a scalable Hadoop Distributed File System** that can support petabytes of data and
  - **a MapReduce Engine** that can computes results in batches.
- **HDFS stores data in the form of block of files and follows the write-once-read-many model to access data from these blocks of files.**
- However, apart from files, different types of database are also used.

## Storage Layer (Contd.)

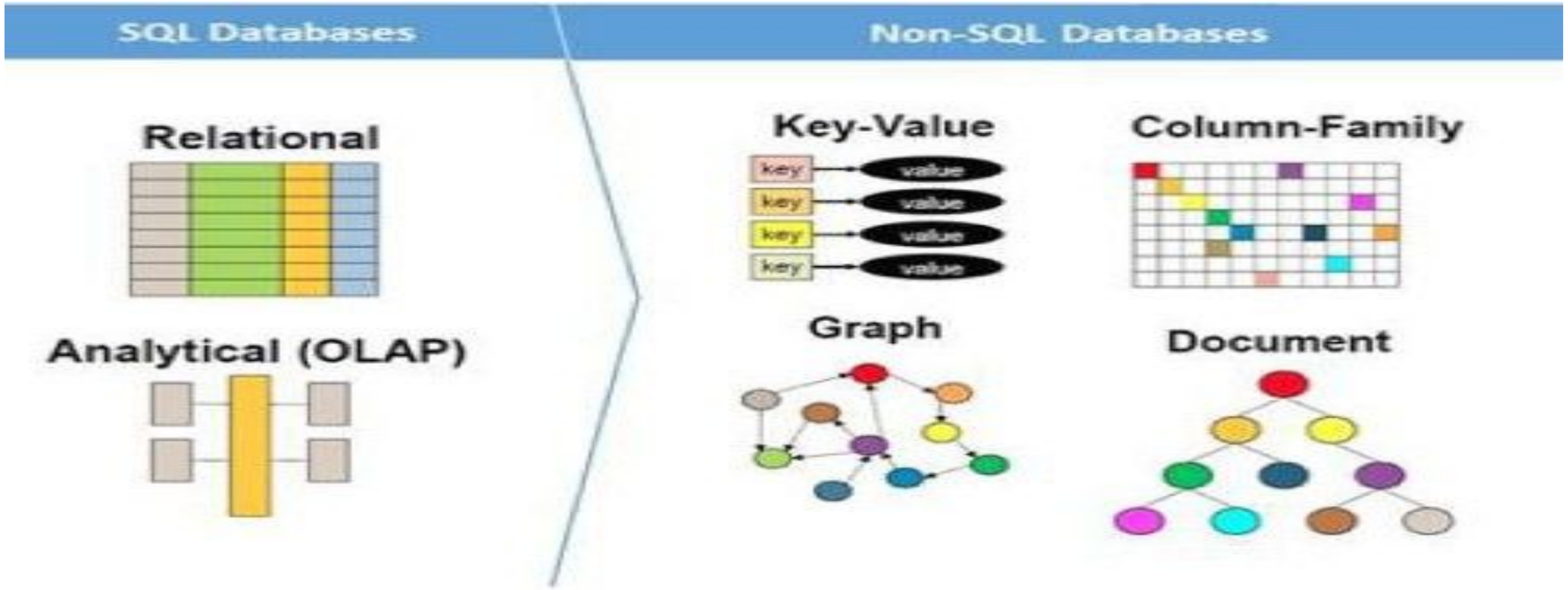
- However, storage requirement can be addressed by a single concept known as Not Only SQL (NoSQL) databases.
- Hadoop has its own database, known as Hbase, but others including Amazon’s DynamoDB, MongoDB, AllegroGraph, Cassandra (used by Facebook) and IndefiniteGraph are popular too.

# SQL Vs. NoSQL

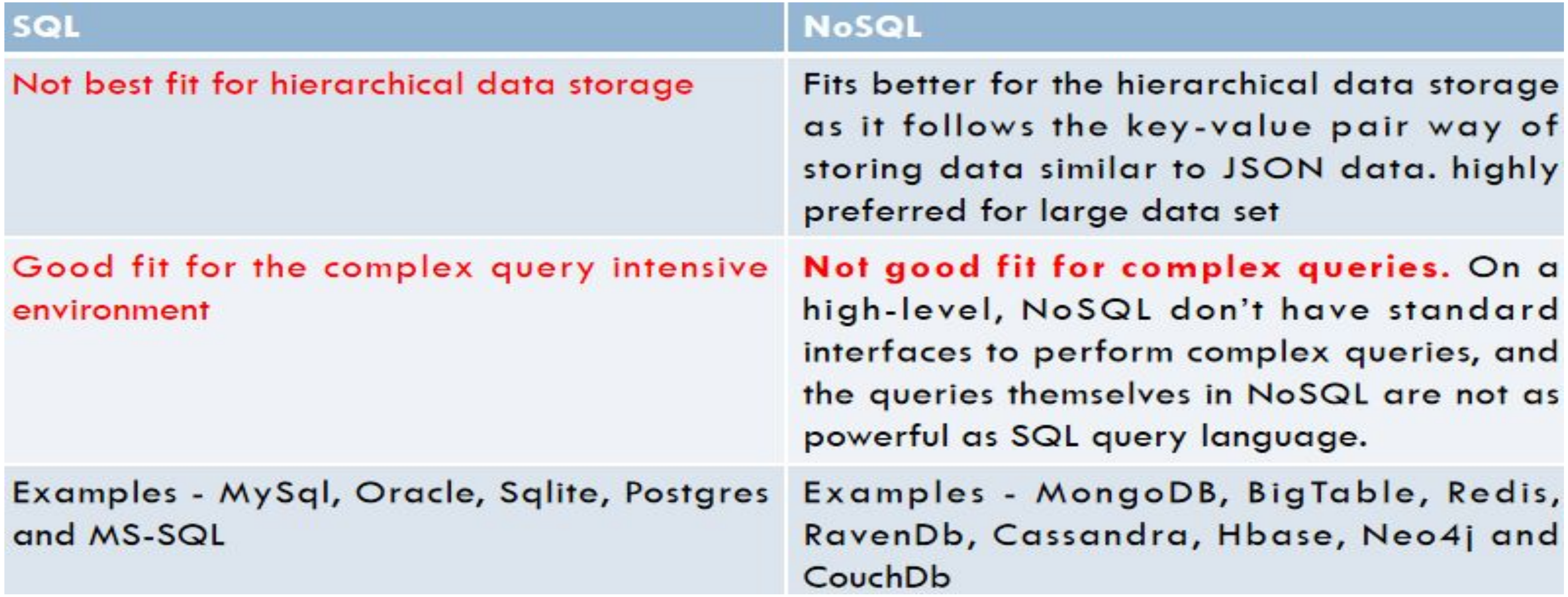


| | NoSQL | SQL |
|---|---|---|
| **Model** | Non-relational | Relational |
| | Stores data in JSON documents, key/value pairs, wide column stores, or graphs | Stores data in a table |
| **Data** | Offers flexibility as not every record needs to store the same properties | Great for solutions where every record has the same properties |
| | New properties can be added on the fly | Adding a new property may require altering schemas or backfilling data |
| | Relationships are often captured by denormalizing data and presenting all data for an object in a single record | Relationships are often captured in normalized model using joins to resolve references across tables |
| | Good for semi-structured, complex, or nested data | Good for structured data |
| **Schema** | Dynamic or flexible schemas | Strict schema |
| | Database is schema-agnostic and the schema is dictated by the application. This allows for agility and highly iterative development | Schema must be maintained and kept in sync between application and database |
| **Transactions** | ACID transaction support varies per solution | Supports ACID transactions |
| **Consistency & Availability** | Eventual to strong consistency supported, depending on solution | Strong consistency enforced |
| | Consistency, availability, and performance can be traded to meet the needs of the application (CAP theorem) | Consistency is prioritized over availability and performance |
| **Performance** | Performance can be maximized by reducing consistency, if needed | Insert and update performance is dependent upon how fast a write is committed, as strong consistency is enforced. Performance can be maximized by using scaling up available resources and using in-memory structures. |
| | All information about an entity is typically in a single record, so an update can happen in one operation | Information about an entity may be spread across many tables or rows, requiring many joins to complete an update or a query |
| **Scale** | Scaling is typically achieved horizontally with data partitioned to span servers | Scaling is typically achieved vertically with more server resources |

## SQL Vs. NoSQL

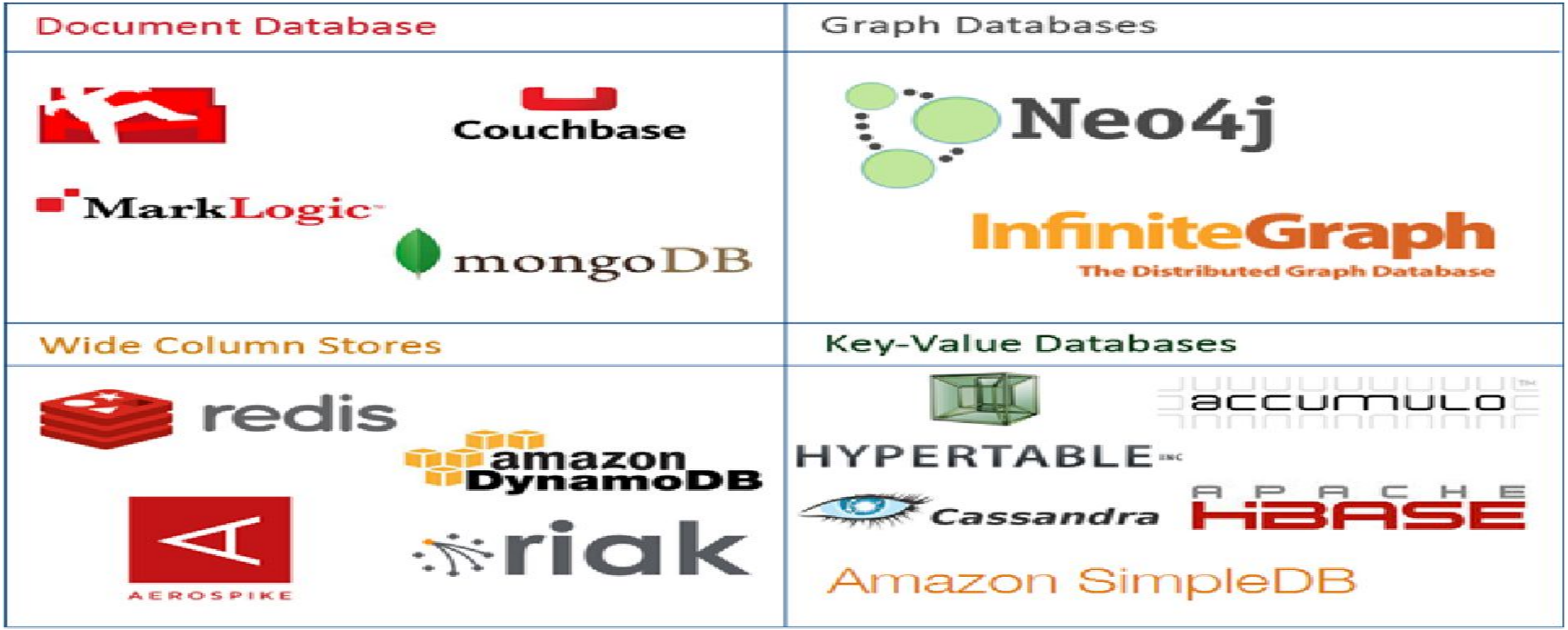

SQL Databases
Non-SQL Databases
Relational
Analytical (OLAP)
Key-Value
key
value
Column-Family
Graph
Document

# SQL Vs. NoSQL (Contd.)



| SQL | NoSQL |
|---|---|
| Not best fit for hierarchical data storage | Fits better for the hierarchical data storage as it follows the key-value pair way of storing data similar to JSON data. highly preferred for large data set |
| Good fit for the complex query intensive environment | **Not good fit for complex queries.** On a high-level, NoSQL don't have standard interfaces to perform complex queries, and the queries themselves in NoSQL are not as powerful as SQL query language. |
| Examples - MySql, Oracle, Sqlite, Postgres and MS-SQL | Examples - MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb |

# Different NoSQL Databases



| Document Database | Graph Databases |
|---|---|
| Couchbase, MarkLogic, mongoDB | Neo4j, InfiniteGraph The Distributed Graph Database |
| **Wide Column Stores** | **Key-Value Databases** |
| redis, amazon DynamoDB, AEROSPIKE, riak | accumulo, HYPERTABLE INC, Cassandra, APACHE HBASE, Amazon SimpleDB |

# References

1. https://www.google.com/url?sa=i&url=https%3A%2F%2Fmedium.com%2Fdigital-transformation-and-platform-engineering%2Fdata-ingestion-processing-and-big-data-architecture-layers-3cb4988c07de&psig=AOvVaw0TExAROjny_PPYqkhxlad4&ust=1691521627600000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCMDpwaCfy4ADFQAAAAAdAAAAABAE
2. https://csveda.com/big-data-architecture-layers/
3. https://medium.com/digital-transformation-and-platform-engineering/data-ingestion-processing-and-big-data-architecture-layers-3cb4988c07de
4. https://www.analytixlabs.co.in/blog/big-data-architecture/
5. https://thorntech.com/sql-vs-nosql/
6. https://community.cloudera.com/t5/Support-Questions/What-is-the-difference-between-SQL-and-NO-sql-Which-one-is/td-p/152336
7. http://www.aryannava.com
8.