Introduction to Big Data

NOSQL DATABASES

Le Ngoc Thanh – Nguyen Ngoc Thao

{lnthanh, nnthao}@fit.hcmus.edu.vn

Outline

- Data repository techniques
- RDBMS vs. NoSQL databases
- NoSQL databases: A categorization

Data repository techniques

The data in an enterprise may come from different sources and in various formats.

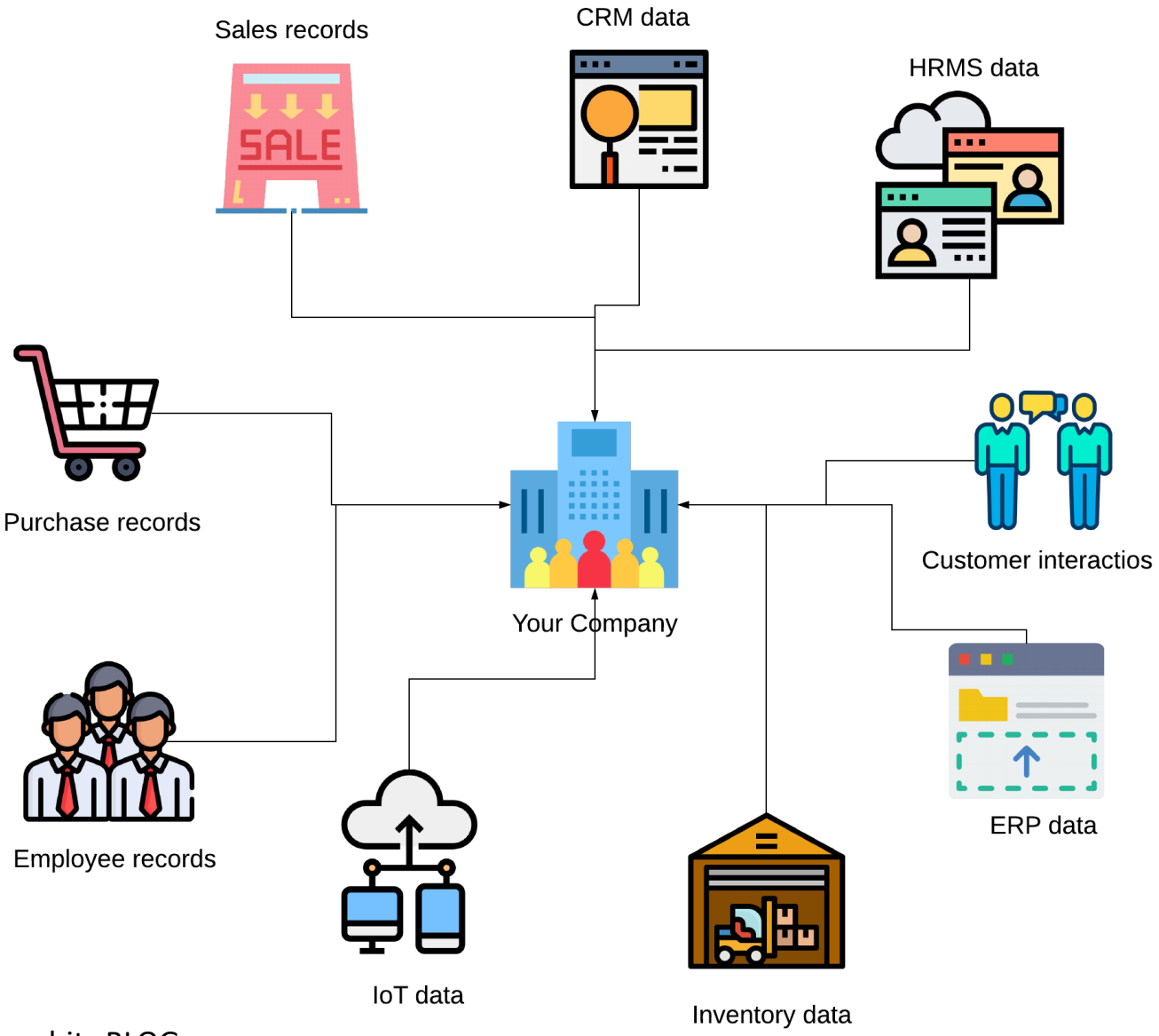
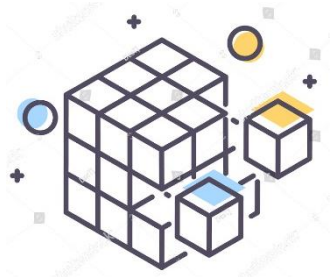


Image credit: accubits BLOG

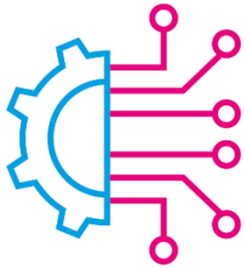
Data warehouse

- A data warehouse is a data management system designed to support analytics and decision makings.
  - It often contain large amounts of historical data derived from a wide range of sources (e.g., log files and transaction applications).

- The key features of a data warehouse include

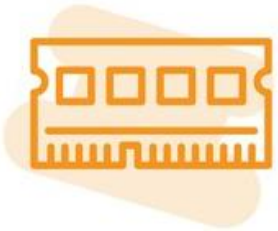


Subject-oriented



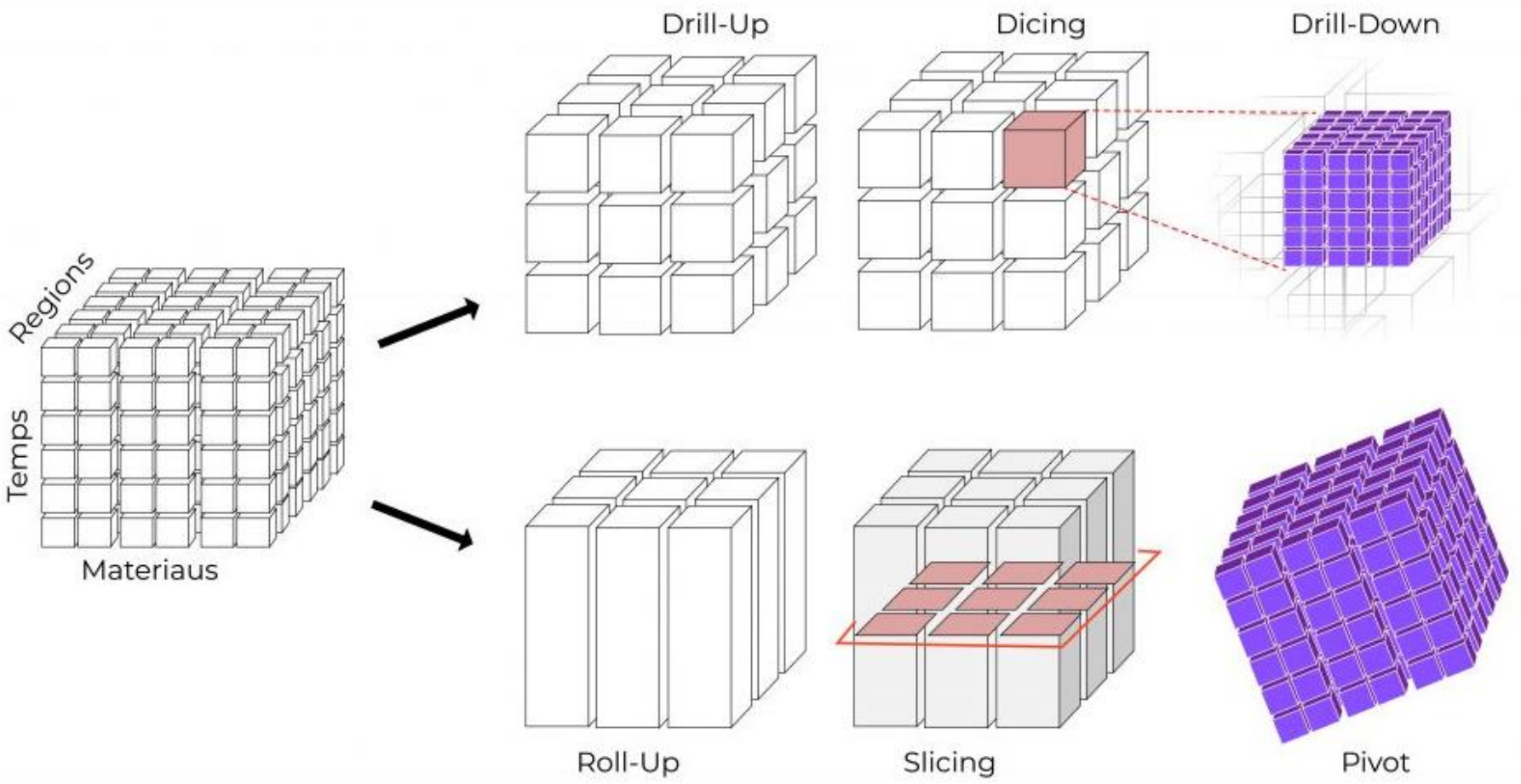
Integrated

Time-variant



Nonvolatile

Data warehouse: OLAP



Basic operations in the **O**n**L**ine **A**nalytical **P**rocess

Image credit: smartboost

Data mart

- A data mart functions similarly to a data warehouse does, but within a much more limited scope.

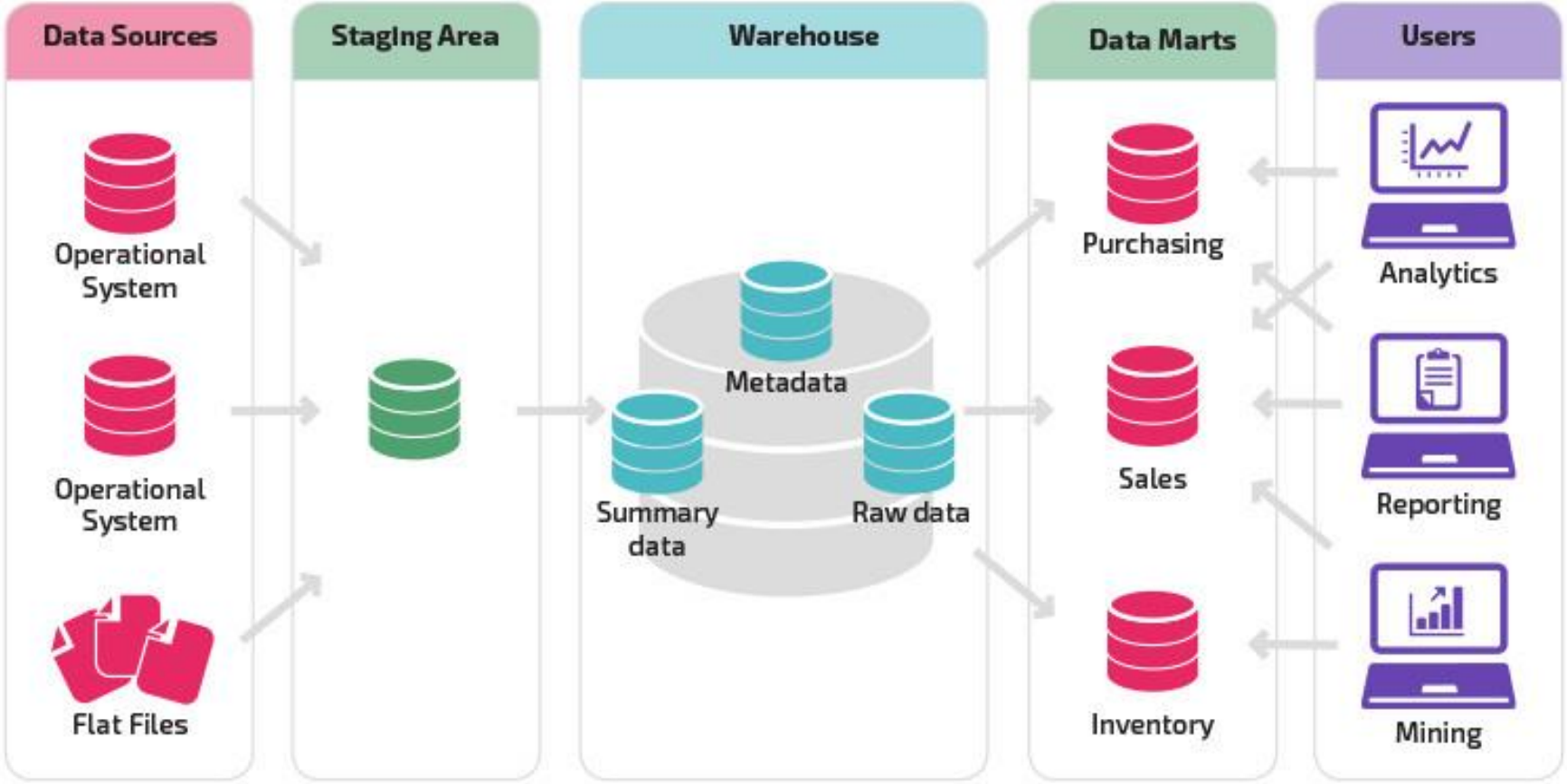


Image credit: Panoply

Data lake

- A data lake is a centralized repository that allows storing all the data at any scale in native format.

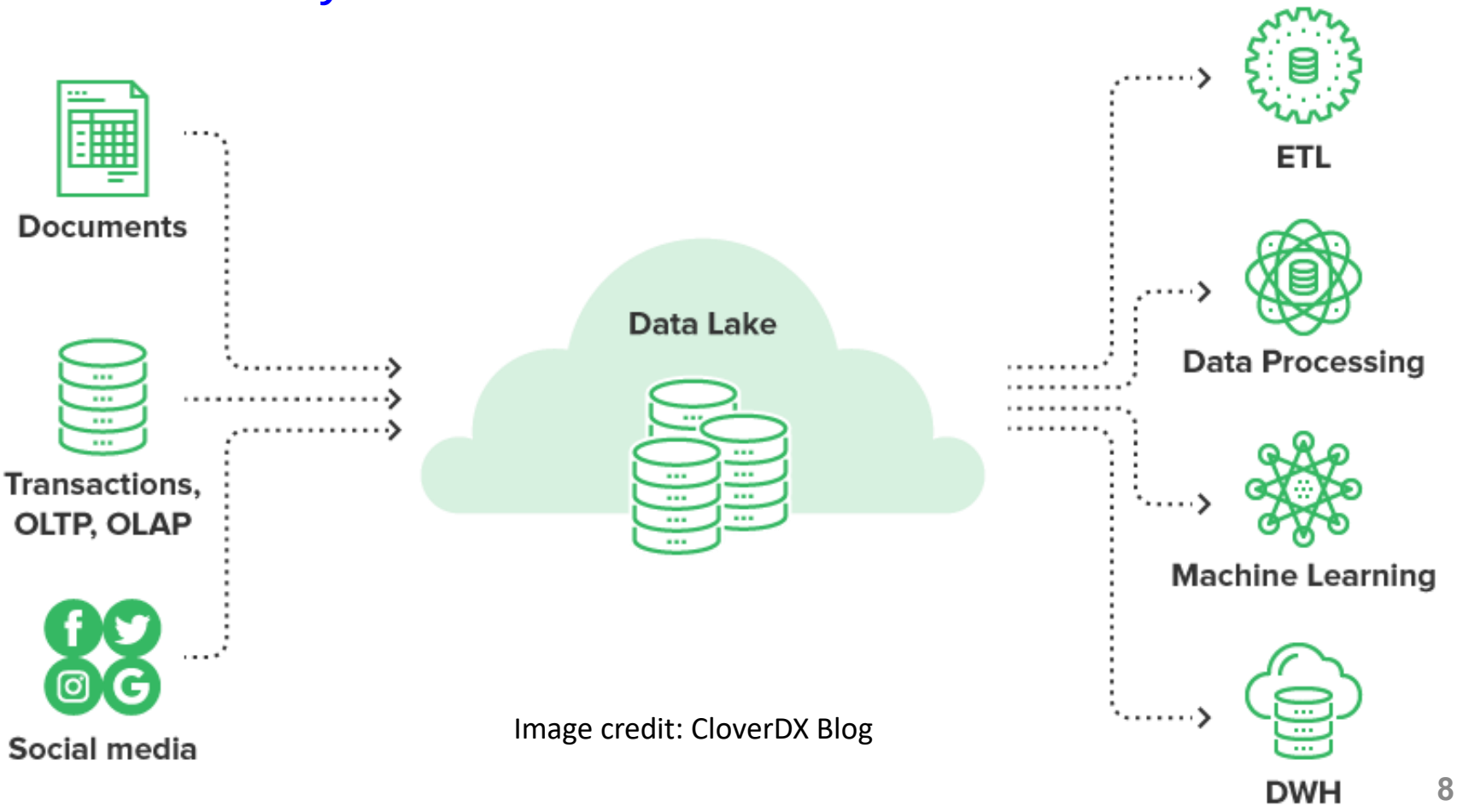


Image credit: CloverDX Blog

Data lake vs. Data warehouse

Characteristics	Data Warehouse	Data Lake
Data	Relational from transactional systems, operational databases, and line of business applications	Non-relational and relational from IoT devices, web sites, mobile apps, social media, and corporate applications
Schema	Designed prior to the DW implementation (schema-on-write)	Written at the time of analysis (schema-on-read)
Price / Performance	Fastest query results using higher cost storage	Query results getting faster using low-cost storage
Data quality	Highly curated data that serves as the central version of the truth	Any data that may or may not be curated (i.e., raw data)
Users	Business analysts	Data scientists, Data developers, and Business analysts (using curated data)
Analytics	Batch reporting, BI and visualizations	Machine Learning, Predictive analytics, data discovery and profiling

Source: Amazon AWS

Data warehouse tools

Data lake tools

An example of unified framework

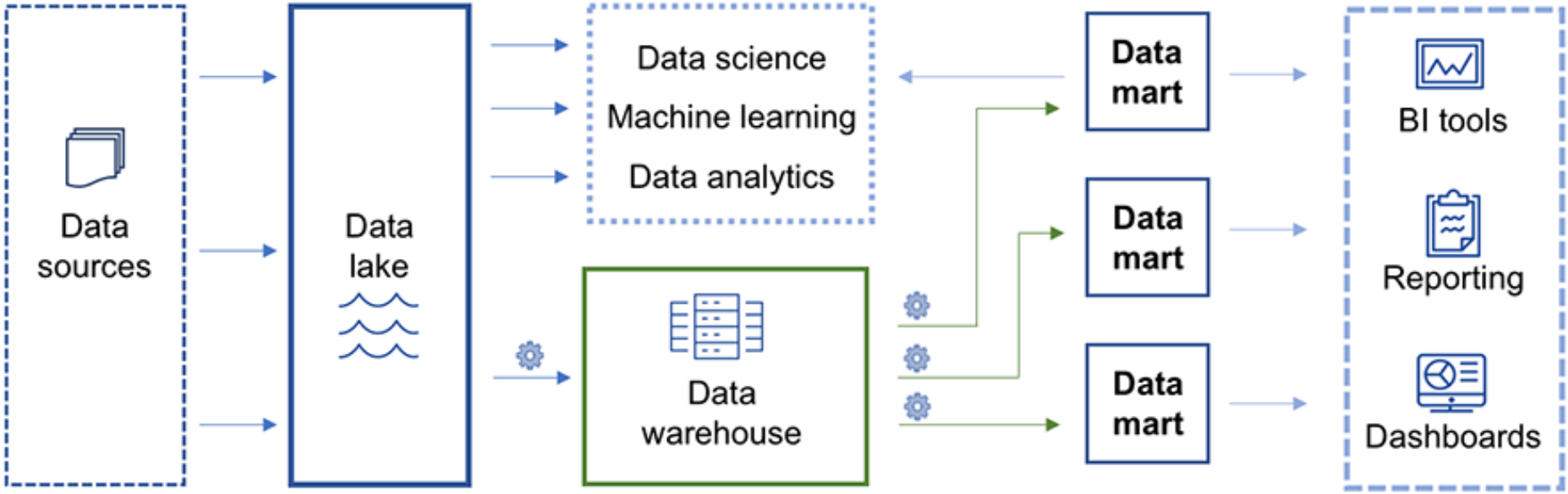
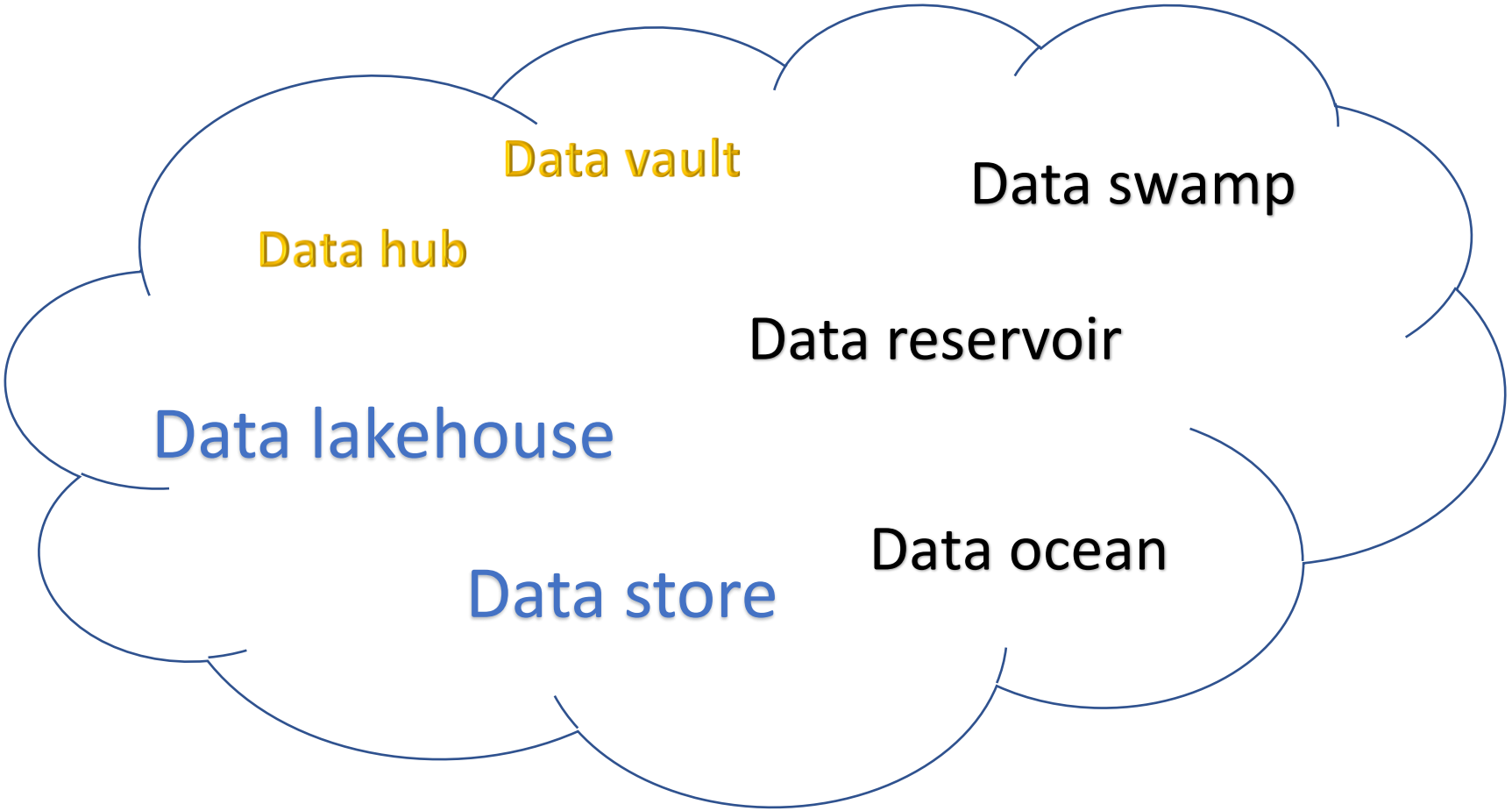


Image credit: Impetus

Have you ever seen these terms? Discover them!



ETL: Extract – Transform – Load

- ETL is the process of moving data from one or multiple data sources to a unified repository.
- Extraction: pull the raw data from different sources
  - Data could come from transactional applications or IoT sensors and it may be in several formats (e.g., relational databases, XML, JSON)
- Transformation: update the data to match organizational needs and data storage solution requirements.
- Loading: data is delivered and secured for sharing, making business-ready data available to internal and external users

ELT: Extract – Load – Transform

- ELT is a variation of ETL in which data is extracted and loaded before it is transformed.

- ELT differs from ETL in the following dimensions.

Transform and Load

Data repository

Supporting tools and communities

The ETL Process

Software Advice.

The ELT Process

Data ingestion

- Data ingestion is the process of driving a wide variety of data sets into where it needs to be in required format and quality.
- ETL and ELT are both viable methods, depending on the business needs.

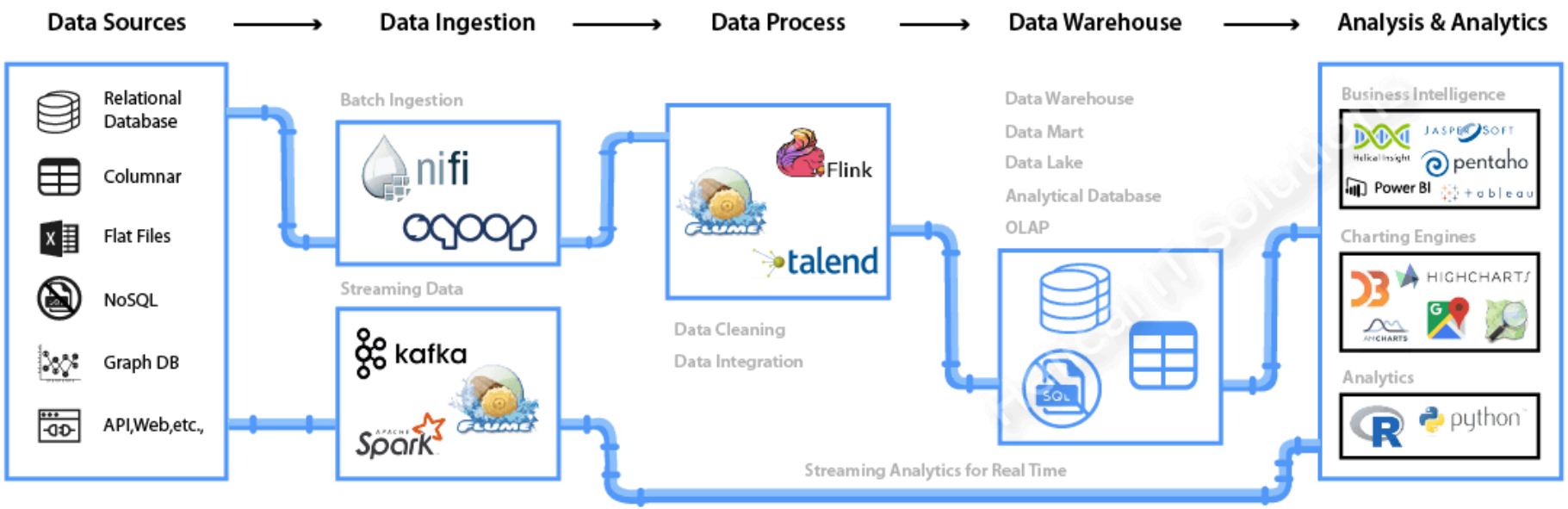
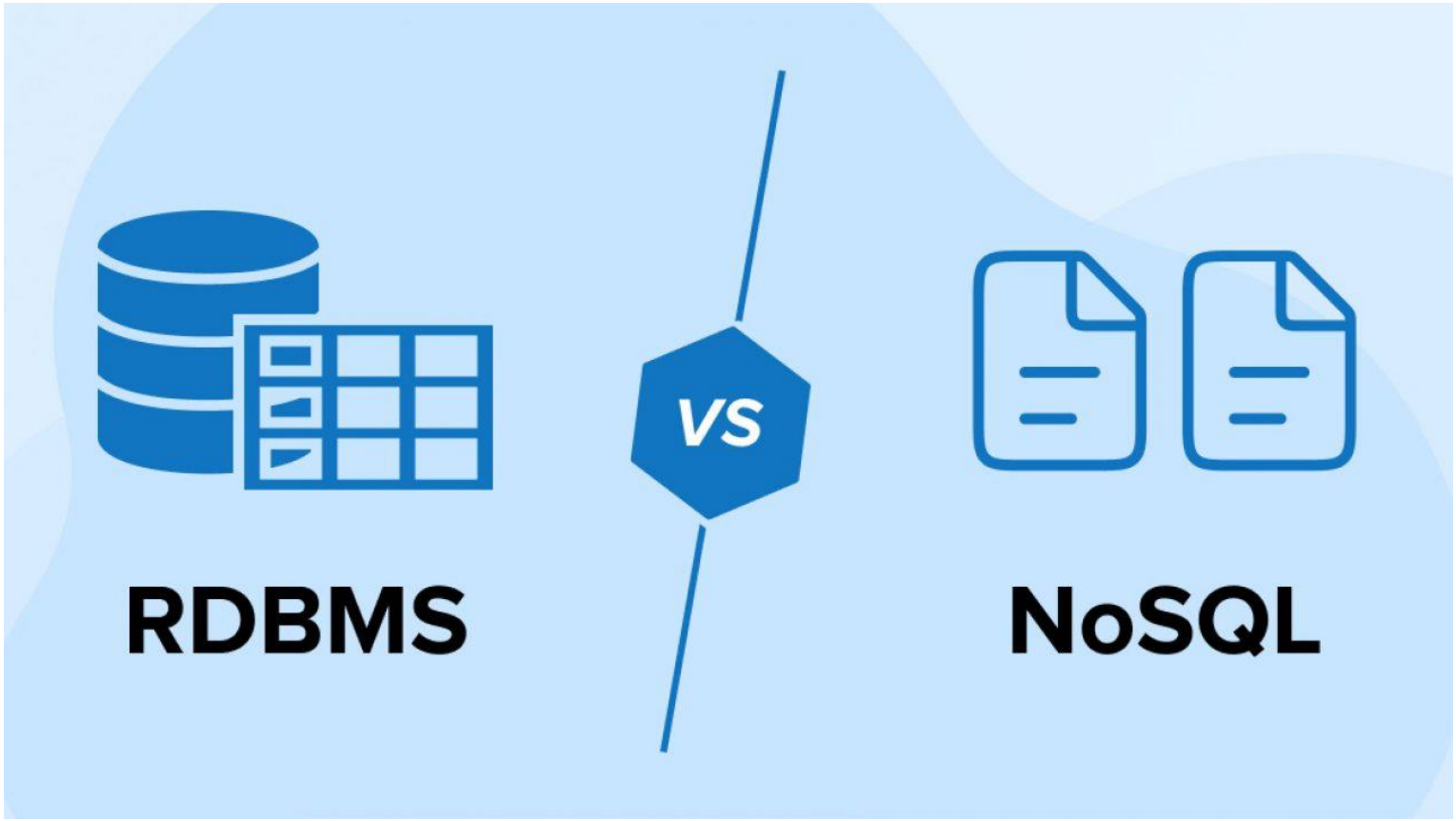
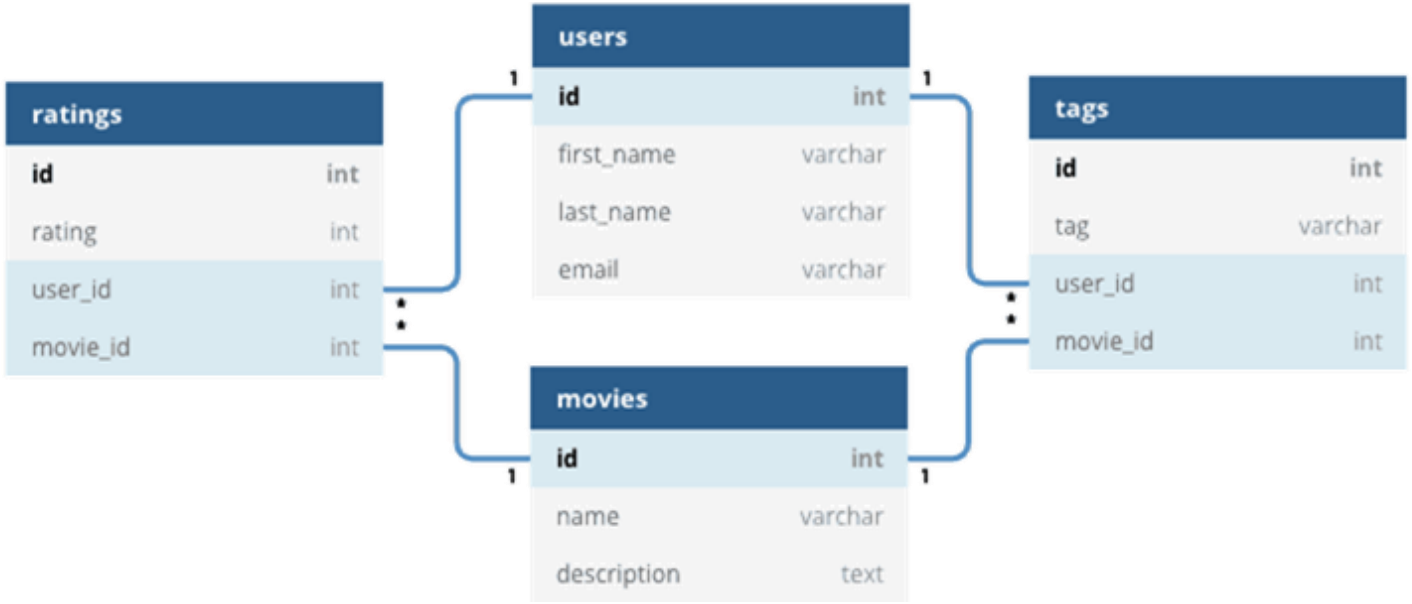


Image credit: Helical IT Solutions Pvt Ltd


VS
RDBMS
NoSQL

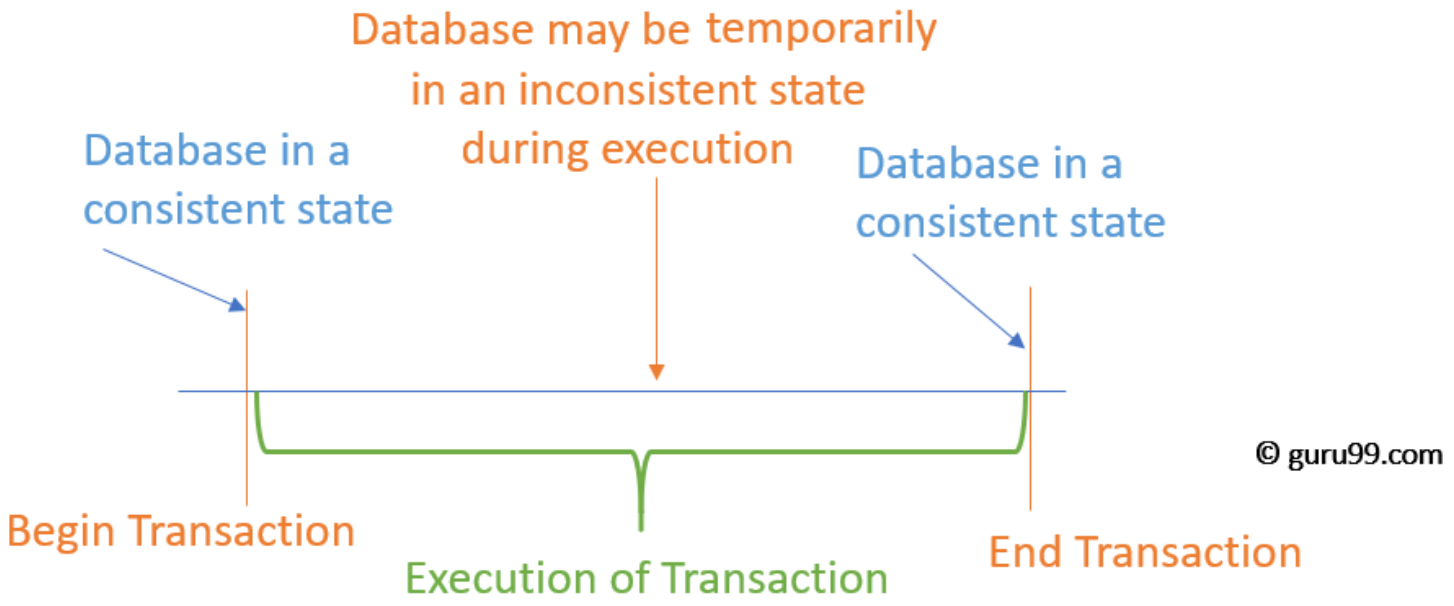
Relational databases

- Relational DB has been a prevalent technology for decades.
  - Mature, proven, and widely implemented
  - Competing database products, tooling, and expertise abound
- Related tables are managed in a strict schema using SQL.
- ACID guarantees are supported.



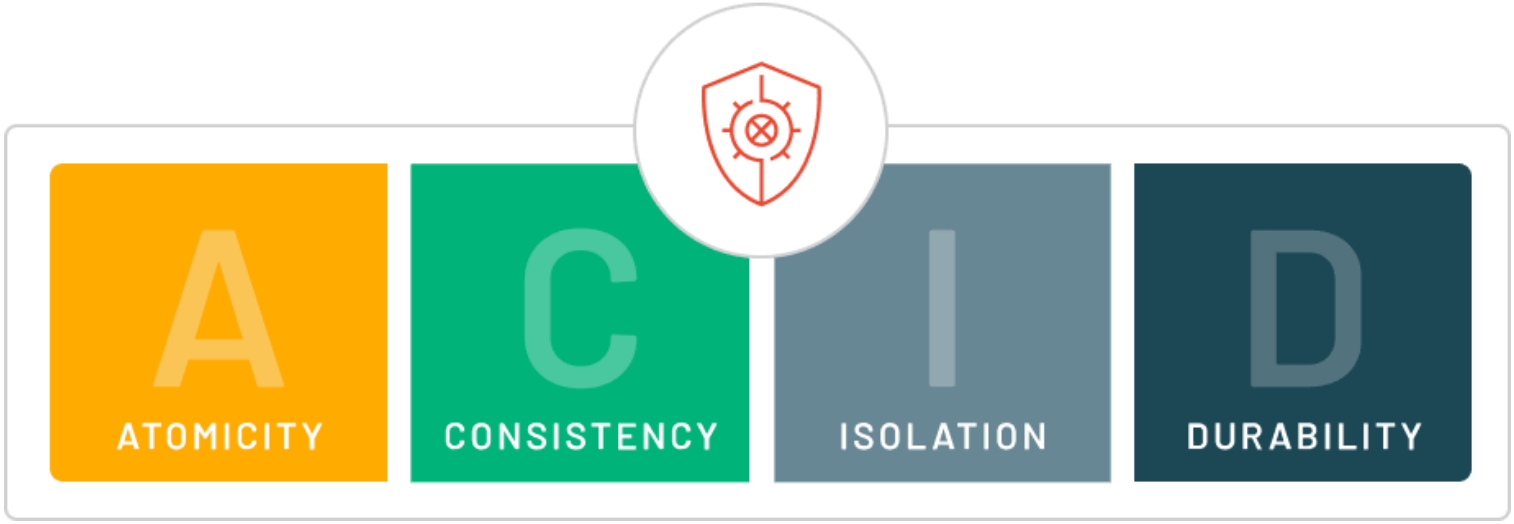
Transactions in RDBMS

- A transaction is any operation treated as a single unit of work, either completes fully or does not complete at all.
  - E.g., withdraw money from bank account: either the money has left your account, or it has not – there cannot be an in-between state
- It leaves the storage system in a consistent state.



The ACID properties

- A set of properties for database transactions that ensure data validity despite errors, power failures, and other mishaps



- Data storage systems that apply these operations are called transactional systems.

Atomicity

- A transaction is a single unit of operation. You either execute it entirely or do not execute it at all. There cannot be partial execution.
- This property prevents data loss and corruption from occurring.

Consistency

- Transactions only make changes to tables in predefined, predictable ways.
- Data errors or corruption do not create unintended effect for the integrity of tables.

Isolation

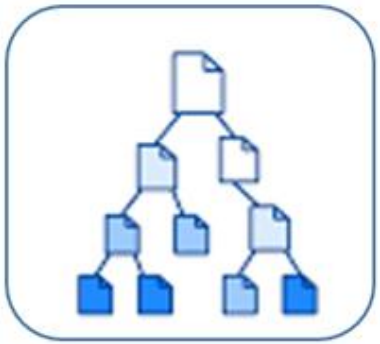
- The concurrent transactions of multiple users do not interfere with another.
- During execution, intermediate transaction results from simultaneously executed transactions should not be made available to each other.

Durability

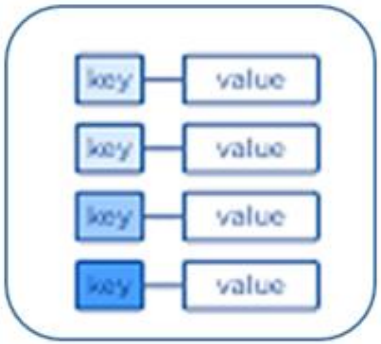
- Changes to data made by successfully executed transactions will be saved, even in the event of system failure.

What are NoSQL databases?

- NoSQL databases are non-tabular since they store data in a format other than relational tables.
- They come in a variety of types based on their data models.

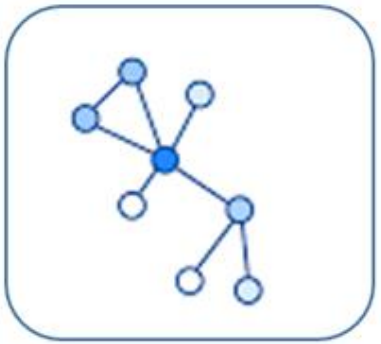


Document Store



Key-Value Store

Wide-Column Store

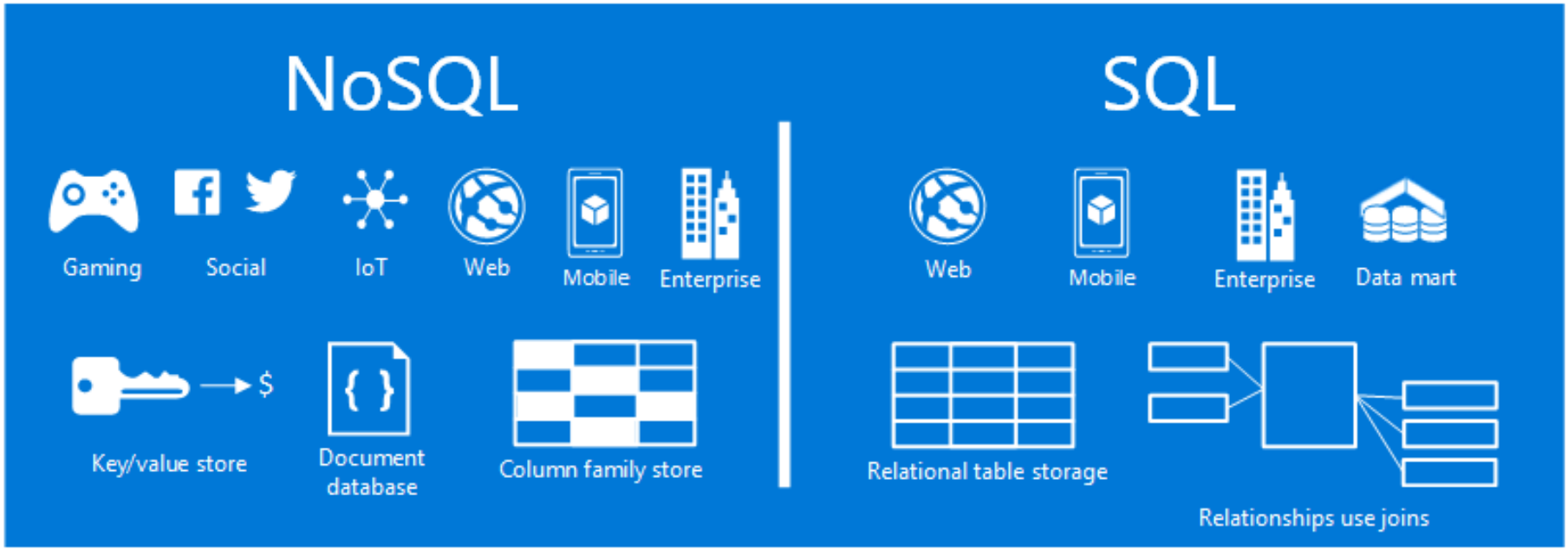


Graph Store

- Provide flexible schemas
- Scale easily with large amounts of data and high user loads

NoSQL databases

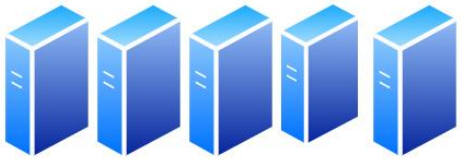
- NoSQL databases excel in their ease-of-use, scalability, resilience, and availability characteristics.
  - Good for high volume services that need sub second response time
- ACID guarantees are typically unsupported beyond the scope of a single database partition.



NoSQL databases: Key features

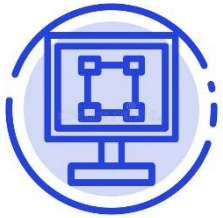
- Each NoSQL database has its own unique features.
- At a high level, many NoSQL DBs have the following features

Flexible schema



Horizontal scaling

Fast queries due to the data model

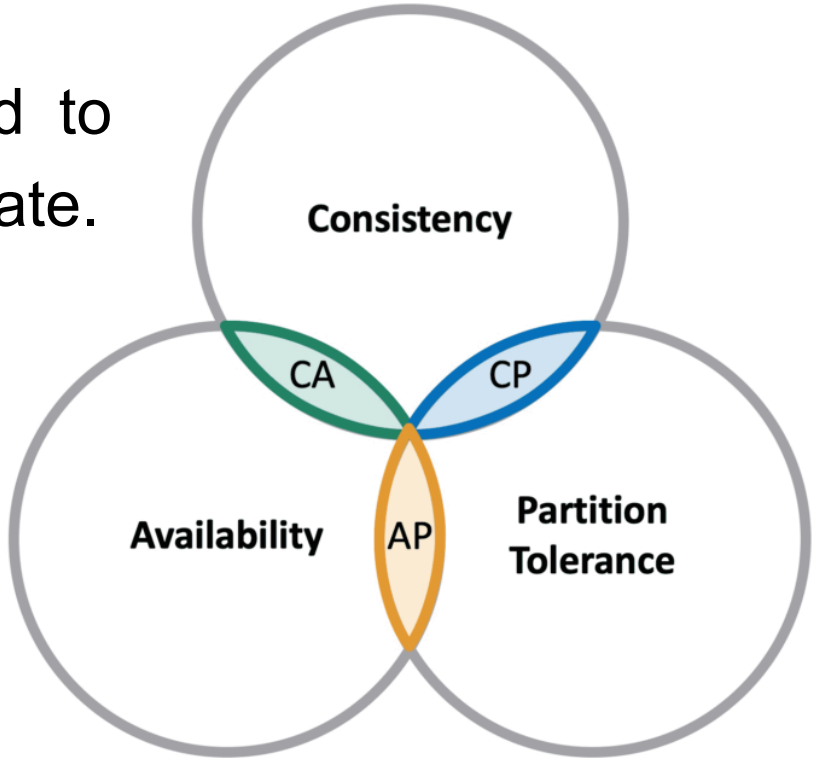


Easy development

Consider a NoSQL datastore when:	Consider a relational database when:
You have high volume workloads that require predictable latency at large scale (e.g. latency measured in milliseconds while performing millions of transactions per second)	Your workload volume generally fits within thousands of transactions per second
Data is dynamic and frequently changes	Data is highly structured and requires referential integrity
Relationships can be de-normalized data models	Relationships are expressed through table joins on normalized data models
Data retrieval is simple and expressed without table joins	You work with complex queries and reports
Data is typically replicated across geographies and requires finer control over consistency, availability, and performance	Data is typically centralized, or can be replicated regions asynchronously
Your application will be deployed to commodity hardware, such as with public clouds	Your application will be deployed to large, high-end hardware

The CAP theorem

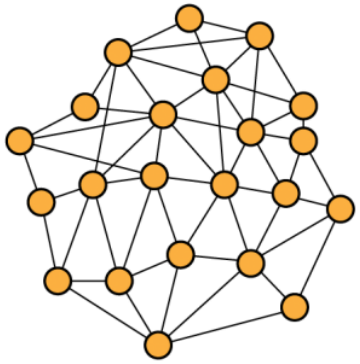
- It is a set of principles applied to distributed systems that store state.



- Any database can only guarantee two of the three properties.
  - E.g., RDBMS (CA), MongoDB (CP), and Cassandra (AP)

C – Consistency

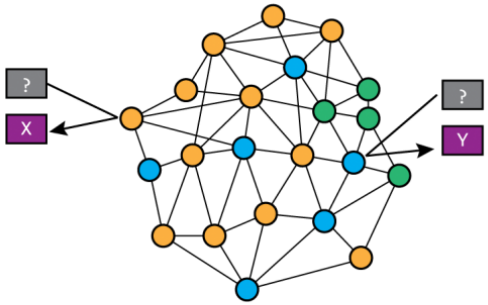
At any given time, all nodes in the network have exactly the same (most recent) **value**.



= Value: X @ 2018-05-03T08:52:40

A – Availability

Every request to the network receives a **response**, without any guarantee that returned data is the most recent.



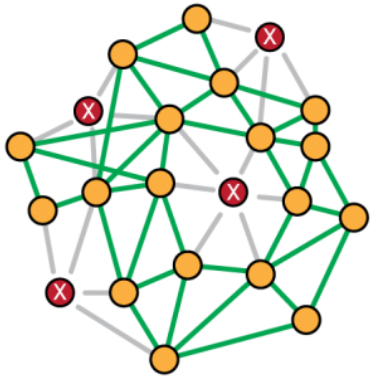
= Value: X @ 2018-05-03T08:52:40

= Value: Z @ 2018-05-03T08:32:58

= Value: Y @ 2018-05-03T07:12:12

P – Partition Tolerance

The network continues to operate, even if an arbitrary number of nodes are **failing**.

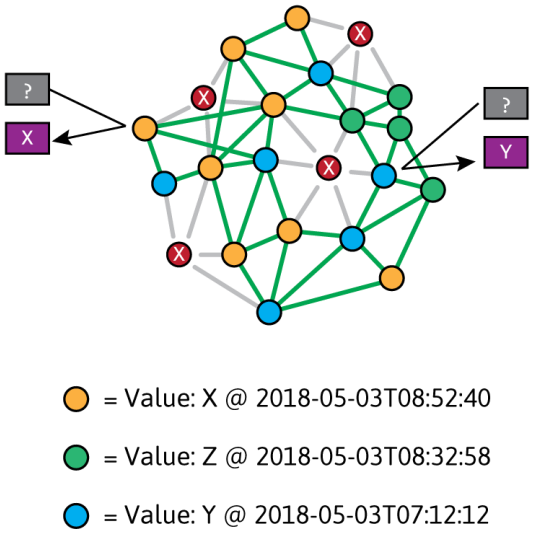


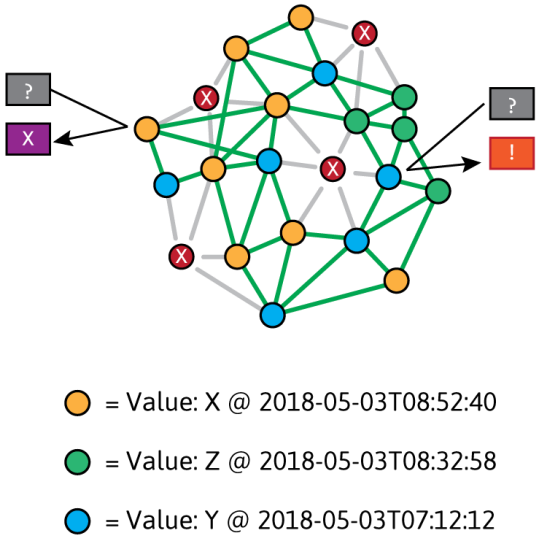
The CAP theorem

- It explains the tradeoffs native to managing consistency and availability during a network partition.
  - However, tradeoffs with respect to consistency and performance also exist with the absence of a network partition.
- Relational databases are CA but not P.
  - They are typically provisioned to a single server and scale vertically by adding more resources to the machine.
- NoSQL databases are typically CP or AP.
  - Partition tolerance can never be ruled out $\rightarrow$ there will always be failing nodes in the network.
  - One must choose between C and A when in the presence of P.

Distributed systems with CAP

- Availability over Consistency (A + P)
  - Every request receives a response, even if the network cannot guarantee it is up to date due to network partitioning (failing nodes).
  - The system will be highly available, yet not guarantee that the data returned is most recent.





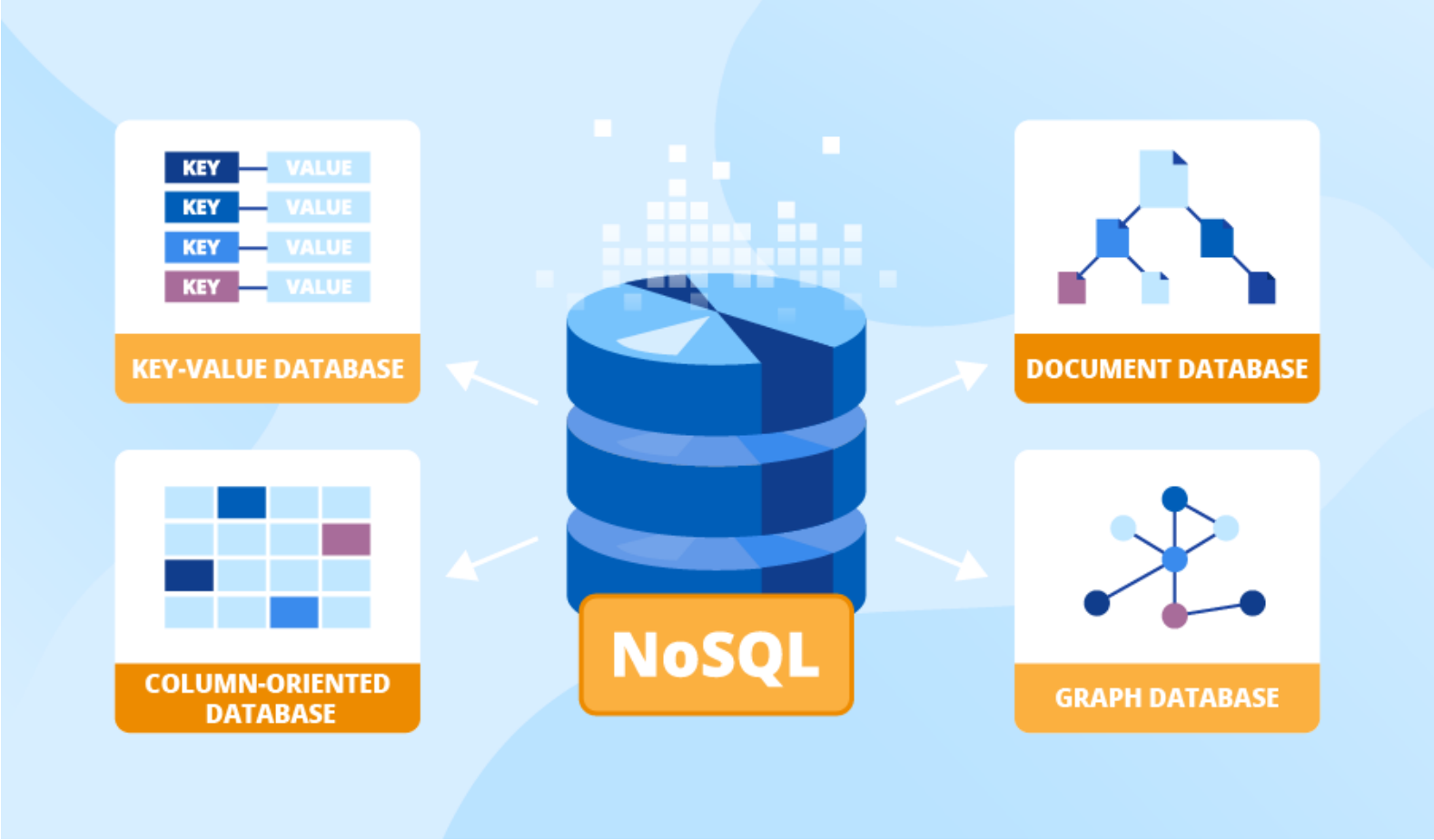
- Consistency over Availability (C + P)
  - The system will return an error or a time-out if particular information cannot be guaranteed to be up to date due to network partitioning.
  - The system will be highly accurate, yet most likely be unavailable for 99.99% of the time.

NoSQL databases: Misconceptions

- NoSQL databases do not store relationship data well.
- They just store relationship data differently than RDBMS do.
- Modeling relationship data in NoSQL databases seems to be easier than in relational databases.
  - The related data does not have to be split between tables, and it can be nested within a single data structure.

NoSQL databases: Misconceptions

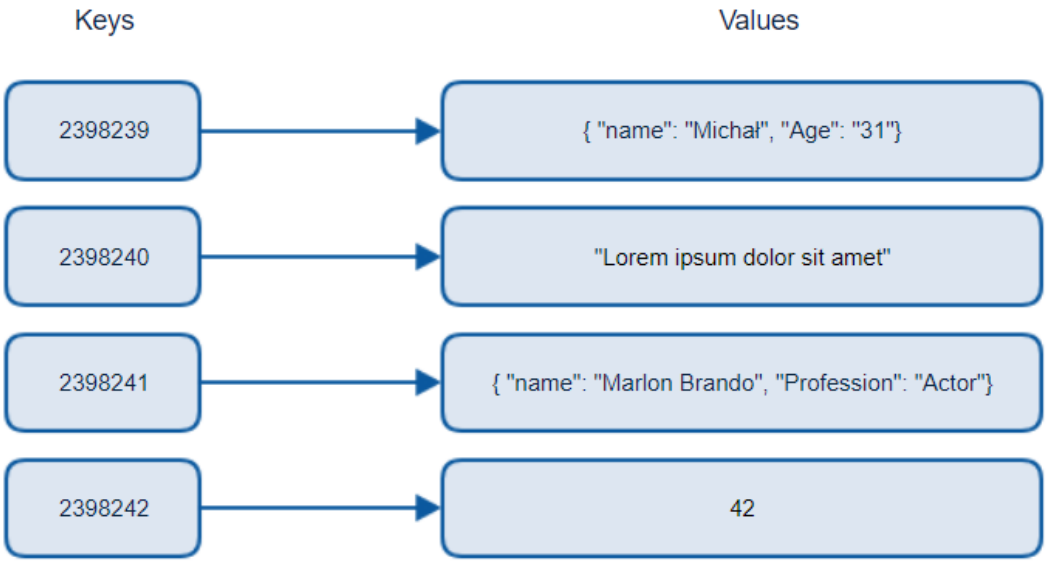
- NoSQL databases do not support ACID transactions.
- Many NoSQL databases do, maybe not fully support.
- The way data is modeled in NoSQL databases can eliminate the need for multi-record transactions in many use cases.
  - Assume that we stored information about a user and his hobbies. Now, we want to update that piece of data.
  - In a relational database, we use a transaction to update records in two tables. In a document one, we only update a single document.



NoSQL databases: A categorization

Data models: Key-value store

- The simplest of the NoSQL databases, data is represented as a collection of key-value pairs.



Keys and values in Azure Cosmos DB

riak

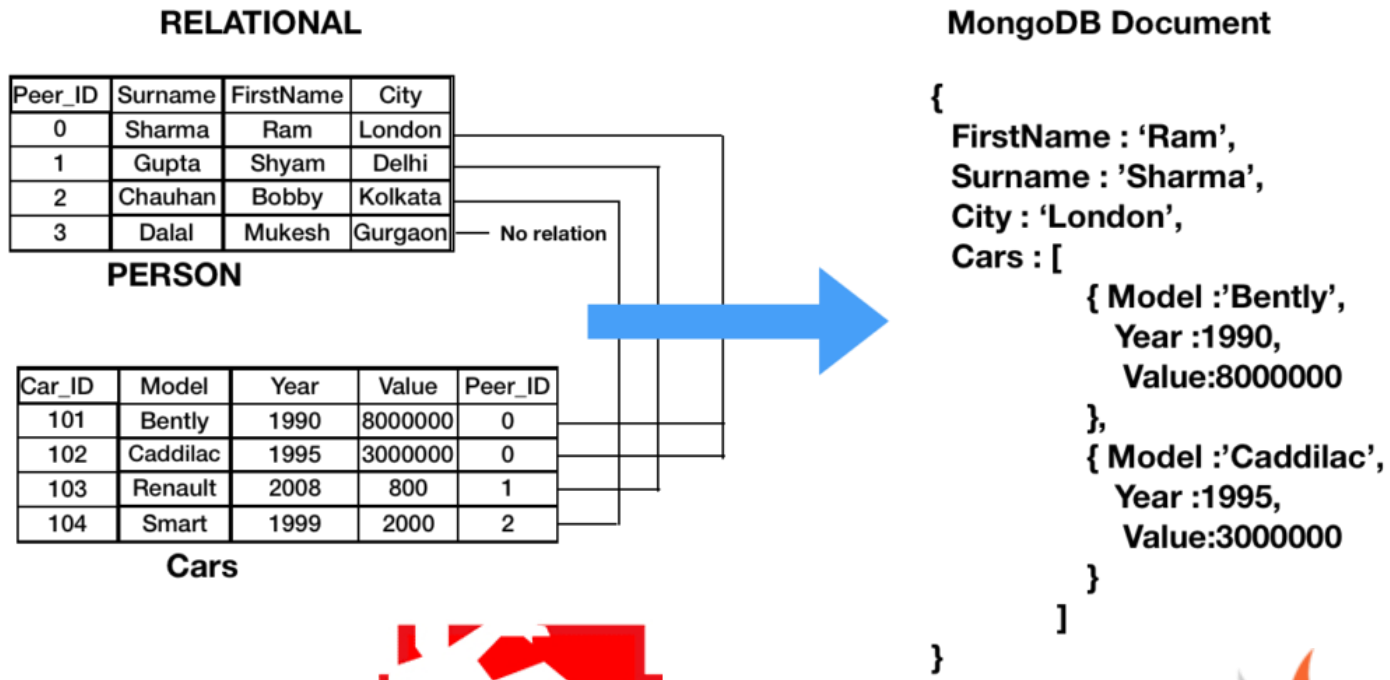
redis

amazon DynamoDB

AEROSPIKE

Data models: Document store

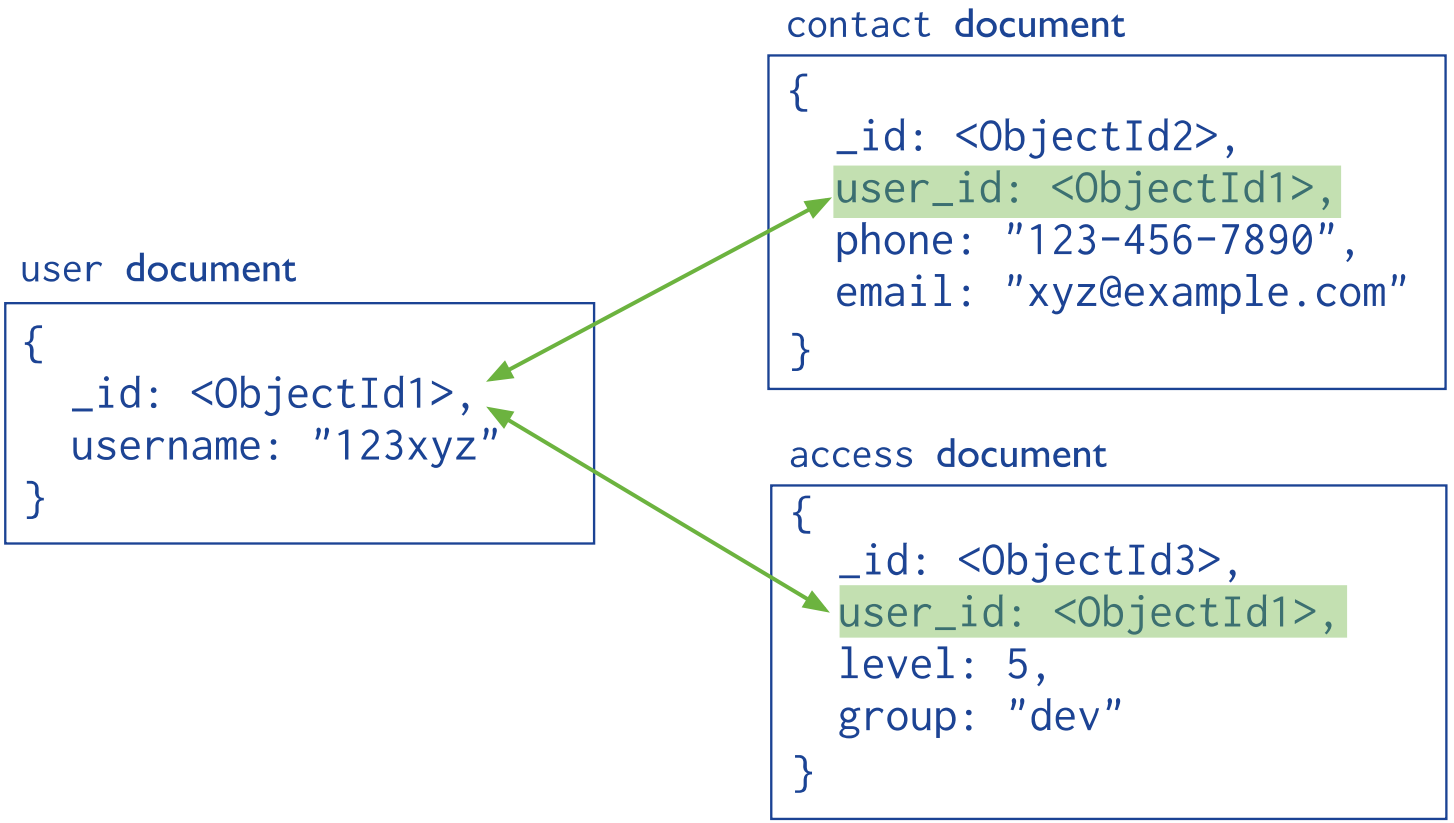
- Data and metadata are stored hierarchically in JSON-based documents inside the database.



mongoDB CouchDB relax elasticsearch OrientDB

Normalized data models

- References between documents describe relationships.



Denormalized data models

- Related data is embedded in a single structure or document.
  - These denormalized data models allow retrieving and manipulating related data in a single database operation.

Data models: Wide-column store

- Related data is stored as a set of nested-key/value pairs within a single column.

Row A	Column 1	Column 2	Column 3	...
	Value	Value	Value	

Row B	Column 2	Column 3	Column 4	...
	Value	Value	Value	

Data models: Graph store

- Data is stored in a graph structure as node, edge, and data properties.

neo4j

JanusGraph

TigerGraph

ArangoDB

AllegroGraph

Multi-model databases

- They are designed to support multiple data models against a single, integrated backend.
  - Document, graph, relational, and key–value models are examples of data models that may be supported by a multi-model database.

	Key-value	Document	Graph	Streaming
Franz Inc. AllegroGraph	☐	☑	☑	☐
ArangoDB	☑	☑	☑	☐
redis	☑	☑	☑	☑

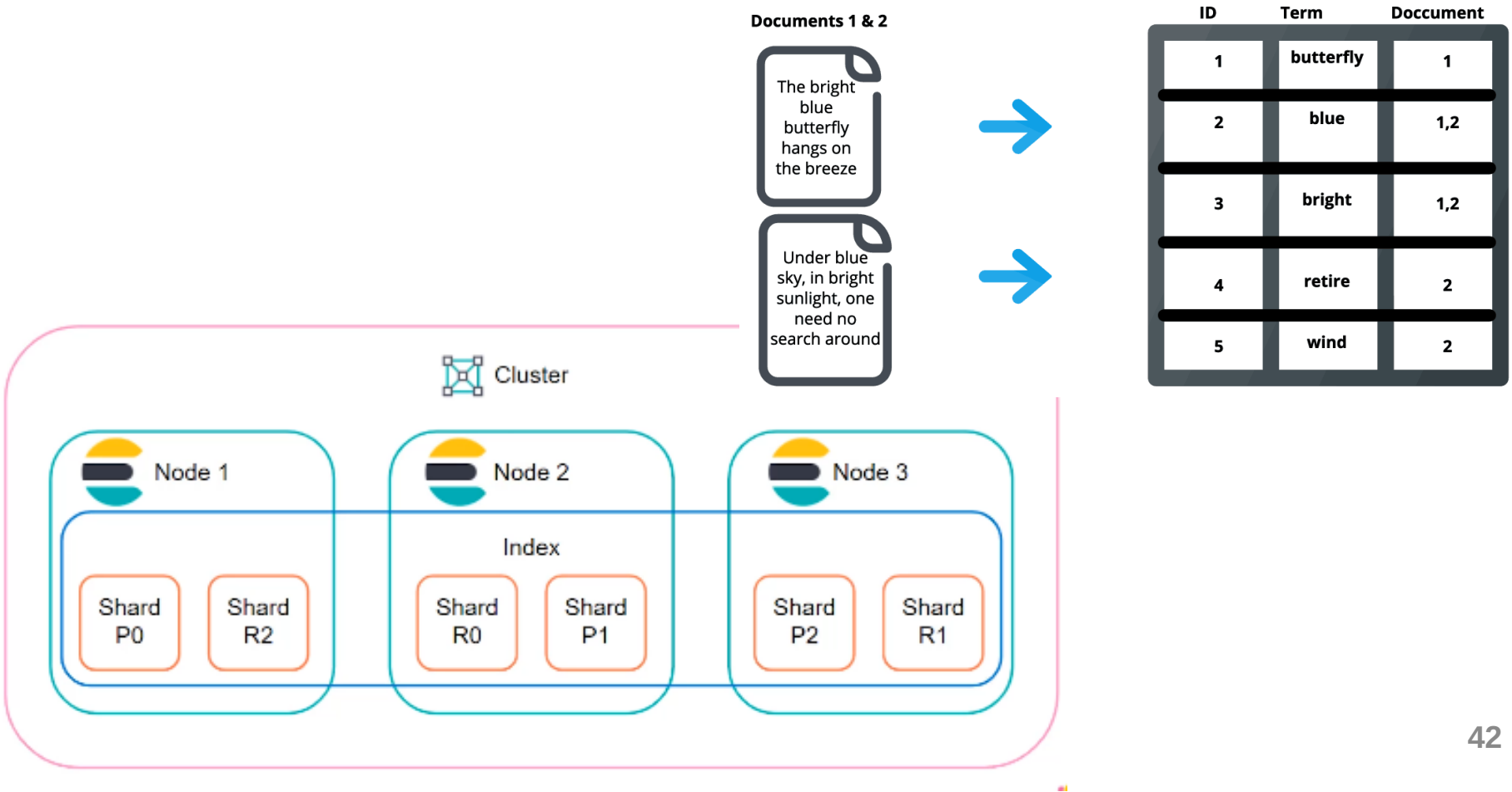
In-memory databases

- They are database management systems that primarily rely on main memory for computer data storage.
  - This contrasts to databases that store data on disk or SSDs
- They are designed to enable minimal response times by eliminating the need to access disks.

A distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents

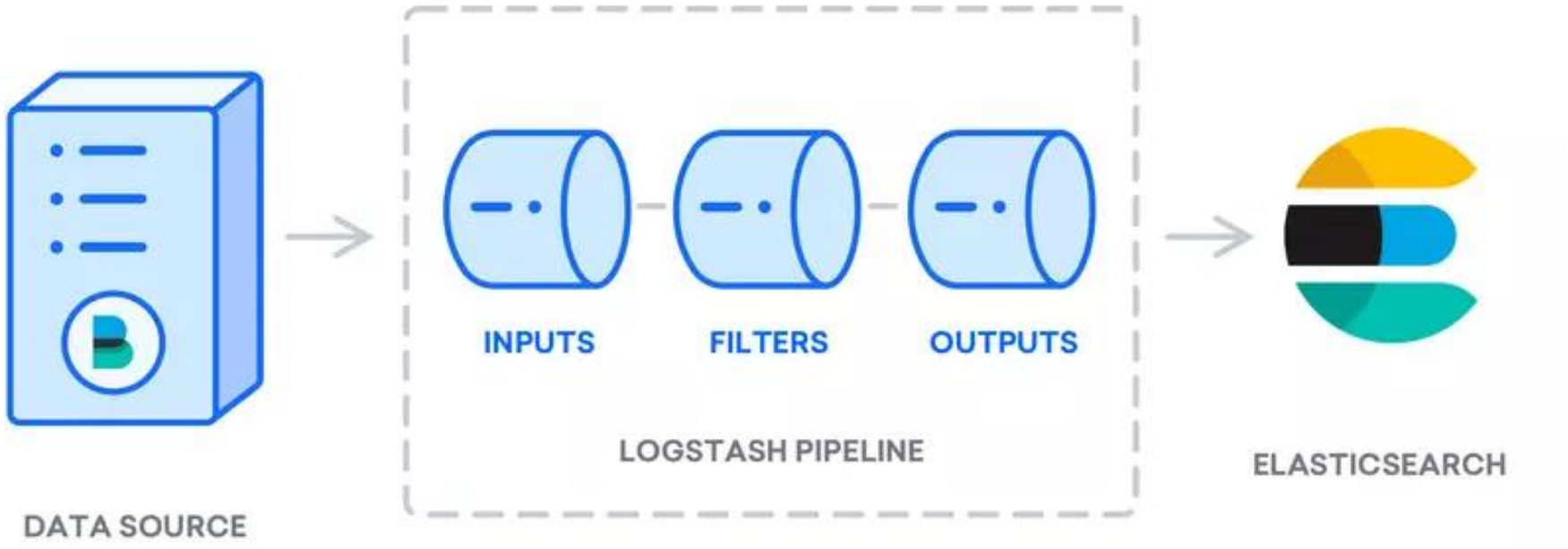
Elasticsearch cluster

- The power of an Elasticsearch cluster lies in the distribution of tasks, searching, and indexing, across all the nodes.



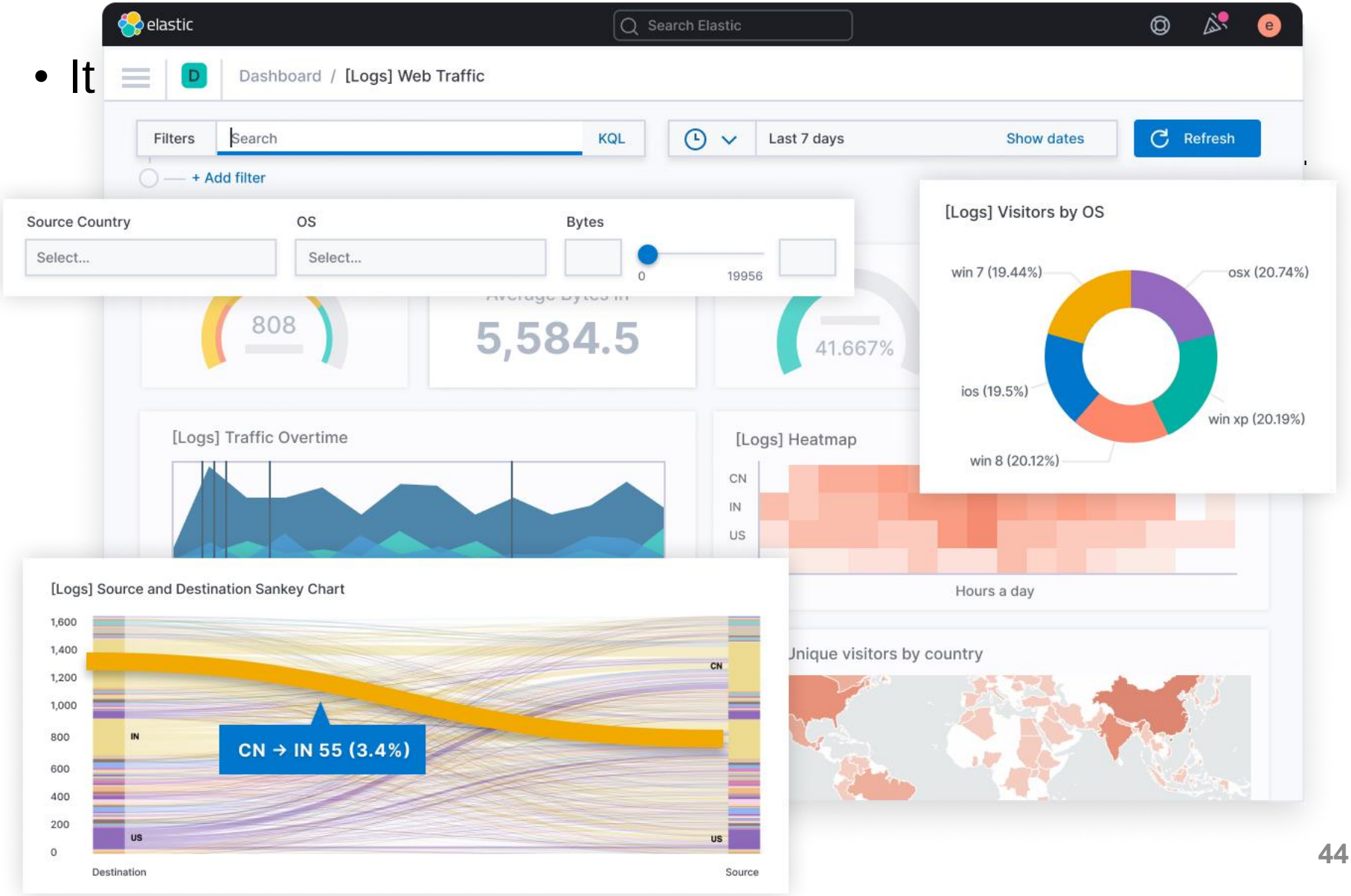
Elasticsearch: Logstash

- It ingests data from a multitude of sources simultaneously, transforms it, and then sends it to Elasticsearch.



Elasticsearch: Kibana

- It




...the end.