reCap - UE BIG DATA ARCHITECTURE APPLIED TO ARTIFICIAL INTELLIGENCE @ ESIGELEC - 2025/2026

Abdel Dadouche - DJZ Consulting

adadouche@hotmail.com

linkedin.com/in/adadouche/

The rise of “modern” BI : Analytics – since Y2K

Business intelligence encompass now a broad category of applications, technologies, and processes for gathering, storing, accessing, analyzing and transforming data into accurate information and support the day to day decision-making processes

BI as a platform for analysis: use cases (for what)

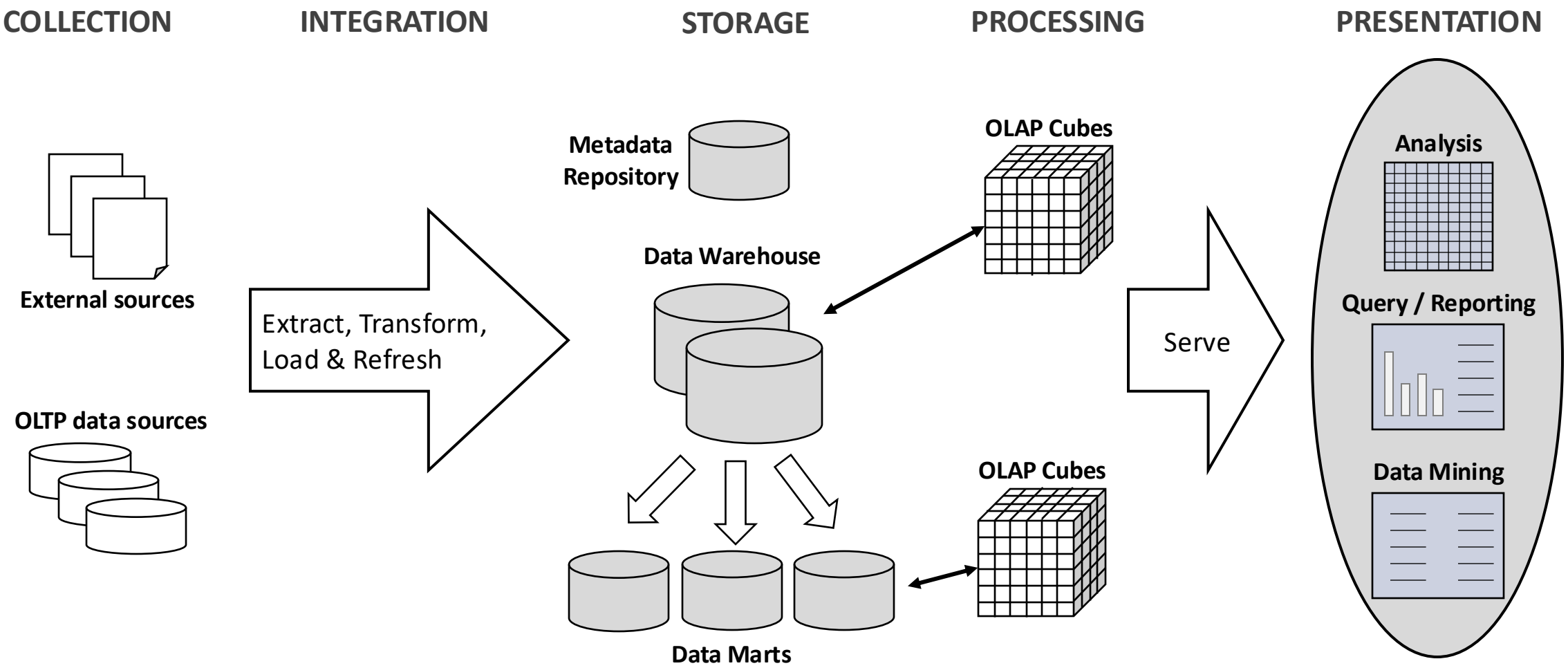
- Accounting : Invoice and order management
- Finance : Cash flow analysis, fraud detection
- HR : Career management, employee satisfaction
- Manufacturing : Production forecasting, cost reduction, quality controls
- Marketing : Campaign analysis
- Sales : Sales forecasting
- …

Inmon definition of a Data Warehouse (1992)

A subject oriented, nonvolatile, integrated, time variant collection of data in support of management's decisions

Source: https://en.wikipedia.org/wiki/Bill_Inmon

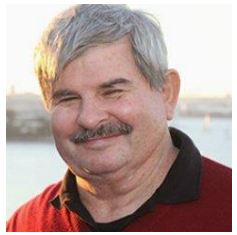
A « Modern » BI architecture



Business Intelligence Process (Wan, J., Chen, T. and Chiu, S. (2005)

Data warehouse vs Operational databases

Features	Operational databases	Data warehouse
Data Granularity	Detailed	Aggregated, summarized, meta data'ed (context)
Data Homogeneity	Homogeneous	Not necessarily homogeneous Data integration often necessary
	Process oriented	Subject oriented
Time variant access	No, only the current view is accessible. Data is periodically archived	Yes, data is partitioned, historized and non volatile
Operations	Many concurrent deletes, inserts and updates and simple reads	Mostly reads (complex queries)



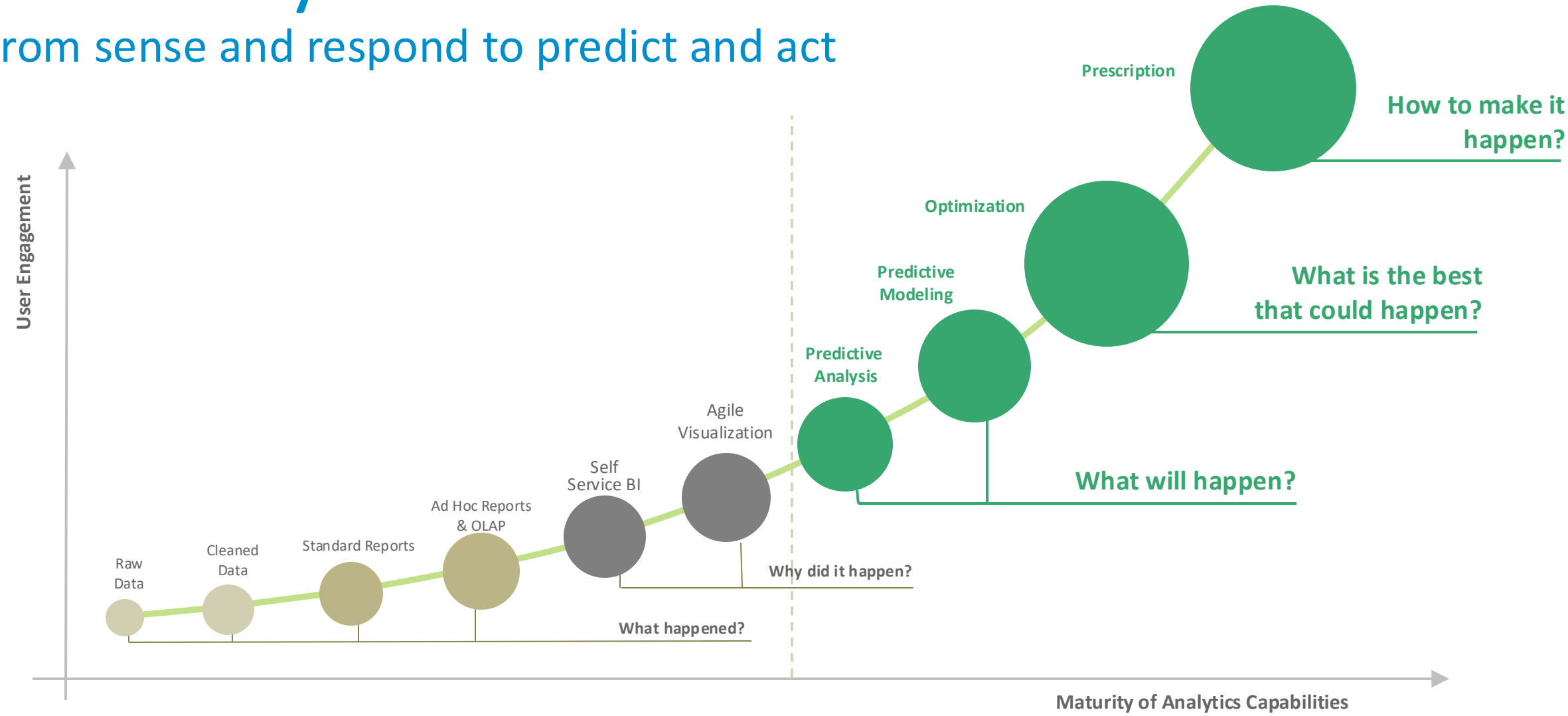
Bill Inmon vs Ralph Kimball

Characteristics	Inmon	Kimball
Business requirement support	Strategic	Tactical
Drivers & Users	Entreprise / Corporate	Business Areas / LOBs
Data structure	Data that meet multiple and varied information needs and non-metric data	KPI, business performance measures, scorecards…
Data sources	Changeable	Stable
Skill sets	Bigger team of specialists	Small team of generalists
Time constraint	Lower / Longer Time Scale	Immediate / Urgent
Cost to build	High start-up costs	Low start-up cost
Budget	Larger	Smaller
Requirements	More stable and growing	Volatile
Modeling approach	3NF	Star / Snowflake

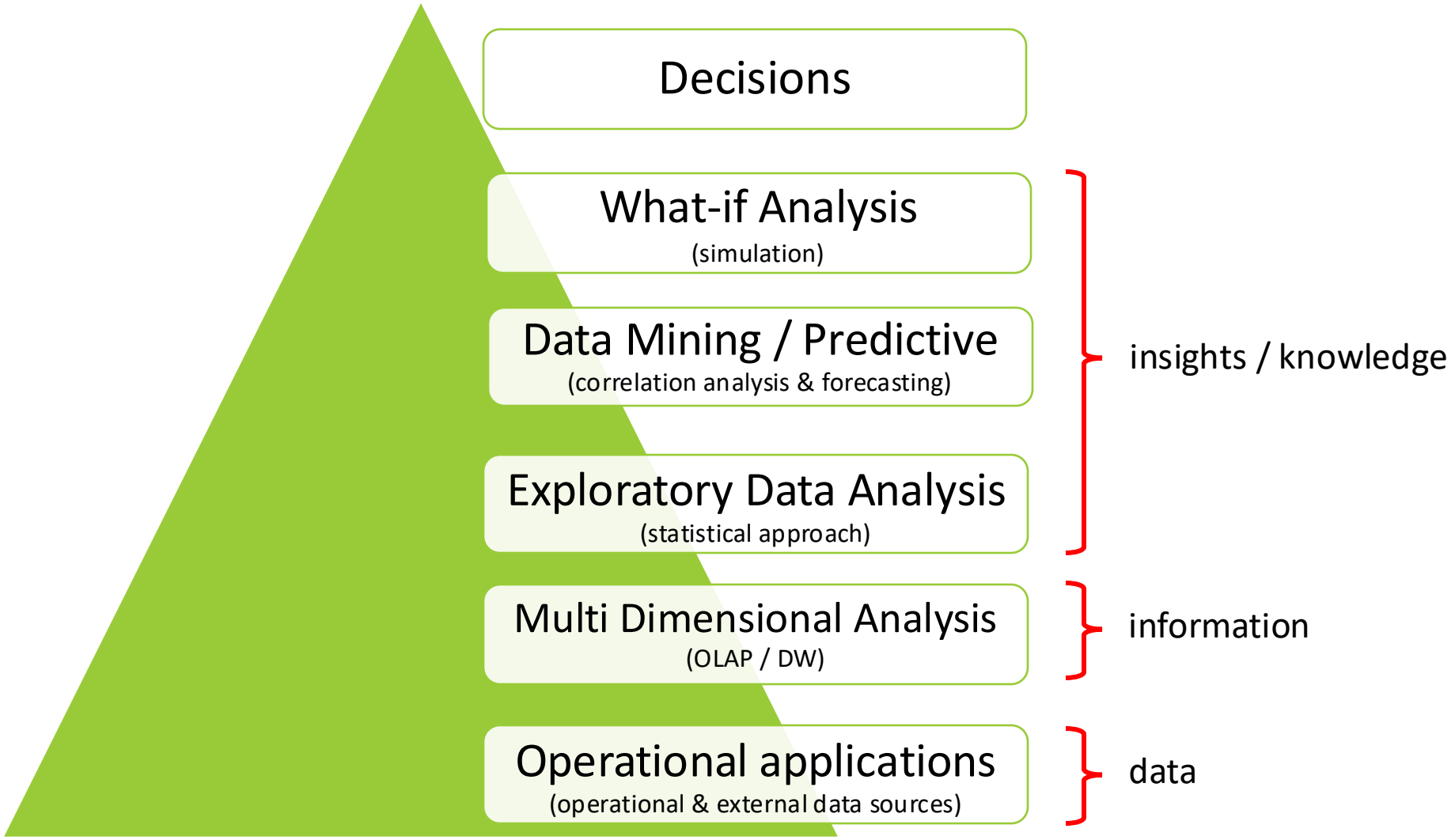
KPI: Key Performance indicators
3NF: 3rd Normal Form

The Analytics Continuum

From sense and respond to predict and act



Modern Analytics Process



OLAP vs OLTP

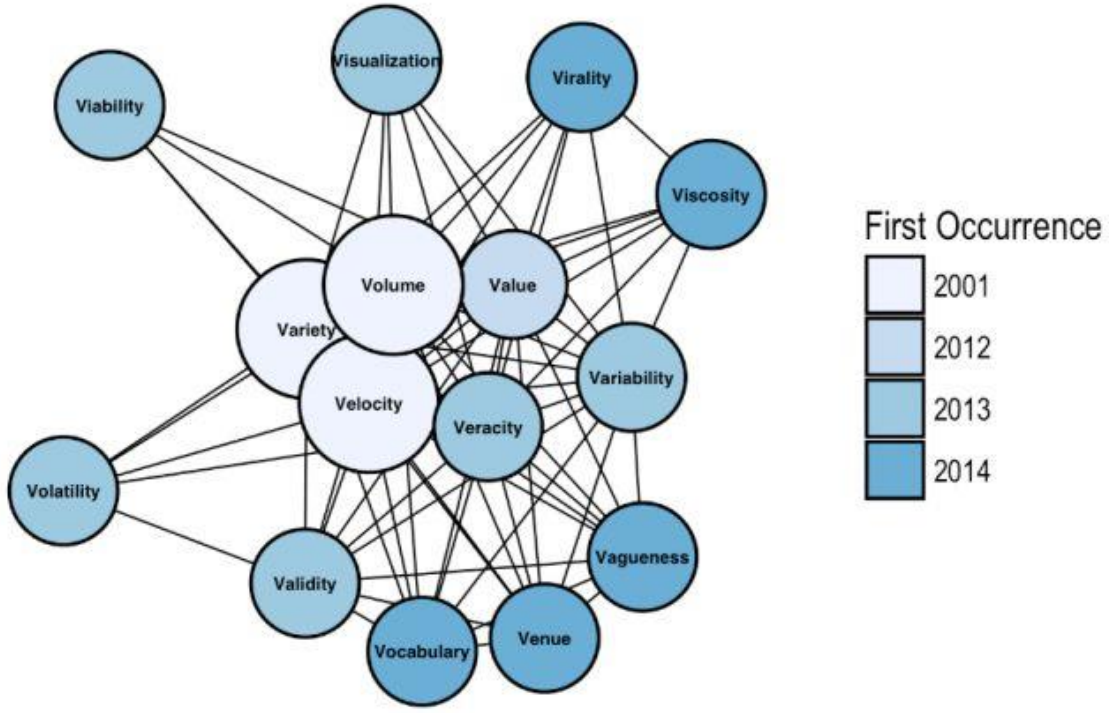
Features		OLTP	OLAP
Concept	**Design**	Transactional	Analytical
	Modelling	Entity-Relation	Star or Snowflake
Data	**Granularity**	Detailed	Aggregated, summarized
	Nature	Relational	Multidimensional
	Updates	Updated and up-to-date	Historical, re-calculated
	Size	GB to TB	TB to PB
Processing	**Unit**	Simple transactions	Complex queries
	Access	Read/Write	Read
	Rows accessed	Ten’s or hundred’s	Million’s or billion’s
	Metric	Transaction throughput	Response time
Users	**Type**	Operator / applications users	Analysts / business users
	Number	Thousand’s	Hundred’s

A History of V's

- The initial 3 V's of Big Data
  - **Volume**: The quantity of generated and stored data
  - **Velocity**: The speed at which the data is generated and processed
  - **Variety** : The type and nature of the data

- Then 7 V's: **Value, Veracity, Variability, Visualization**
- And some more: **Validity, Vulnerability, Volatility**

Now, up to the 42 V's of Big Data & Data Science by Tom Shafer, Elder Research, Inc.



Source: https://en.wikipedia.org/wiki/Big_data
https://www.kdnuggets.com/2017/04/42-vs-big-data-data-science.html

CAP Theorem Overview

In **theoretical** computer science, the CAP theorem states that it is impossible for a distributed data store to simultaneously guarantee more than two out of the following three:

- **C**onsistency:
  - All nodes should see the same data at the same / all time
  - So every read must receive the most recent write or an error
- **A**vailability:
  - Node failures do not prevent survivors from continuing to operate
  - So every request receives a (non-error) response, but without the guarantee that it contains the most recent write
- **P**artition-tolerance:
  - The system continues to operate despite network partitions
  - So every request receives a (non-error) response even if an arbitrary number of messages are being dropped (or delayed) by the network between nodes

Some product examples

- AP
  - Amazon DynamoDB: Read-repair, application hooks
  - Apache Cassandra: Partitioning, Read-repair
  - Apache CouchDB
- CA:
  - Most RDBMS (MySQL, PostgreSQL…), Greenplum, Vertica
- CP:
  - Apache HBase
  - MongoDB
  - Redis
  - Google BigTable

Comparing RDBMS, Data Warehouse, Data Lake, and Lakehouse

Feature	RDBMS	Data Warehouse	Data Lake	Lakehouse
Data Types	Structured	Structured	All (structured + raw)	All (structured + raw)
Data Format	**Closed** proprietary format	**Closed** proprietary format	**Open** format	**Open** format
Schema	Schema-on-write	Schema-on-write	Schema-on-read	Both
Workload Optimized For	OLTP	OLAP	Big data & ML	BI + ML/AI
ACID Transactions	Yes	Limited	No	Yes
Scalability & Cost	Moderate	Expensive	Highly scalable, low-cost	Scalable, cost-effective

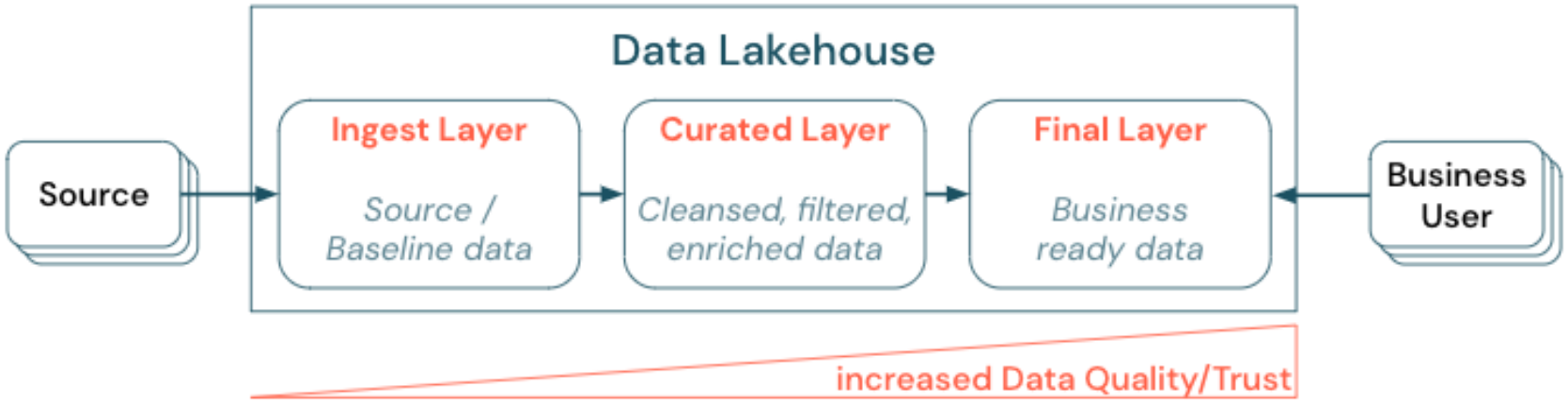
Source : https://www.qlik.com/us/data-lake/data-lakehouse

References

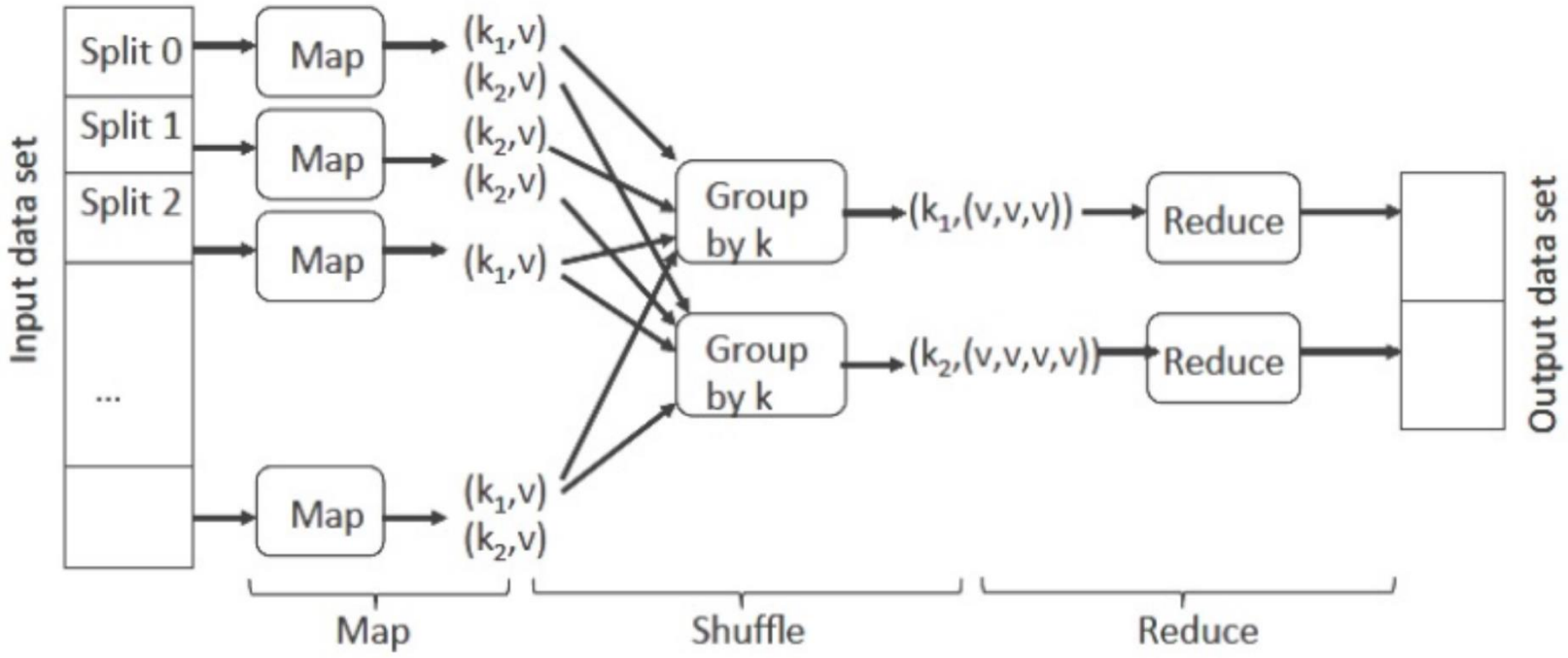
- Inmon, W.H. (2005) Building the Data Warehouse.
- Kimball, R. (2013) The Data Warehouse Toolkit.
- Stonebraker, M. (2024). What Goes Around Comes Around... And Around...?
- Azure Data Lake
- Databricks (2020). The Lakehouse Architecture.
- Google Cloud BigQuery

The medallion architecture?

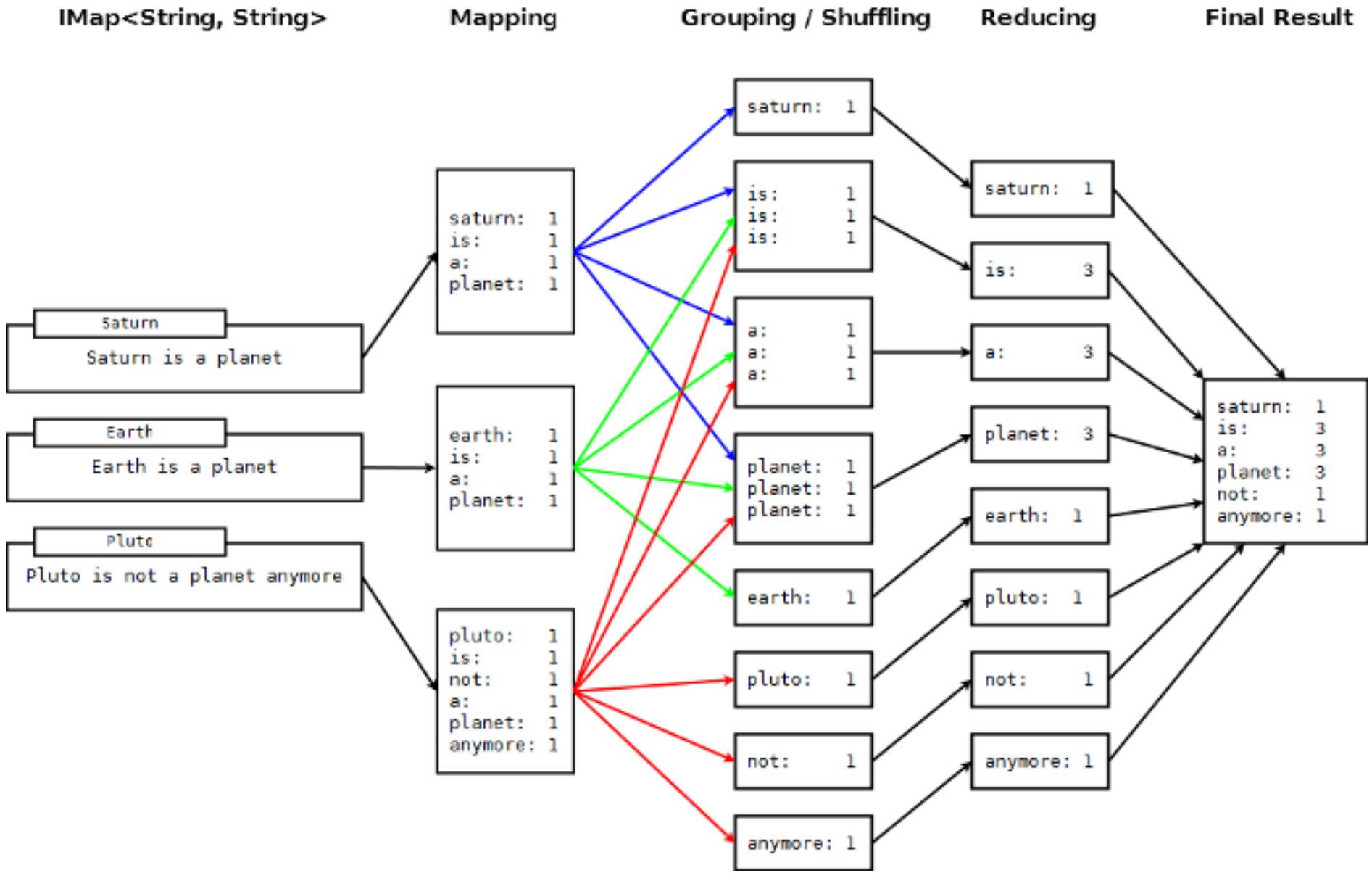
- Define each layer
  - the type of use
  - by who
  - for what purpose
  - what type of schema (3NF, Star, Snowflake)



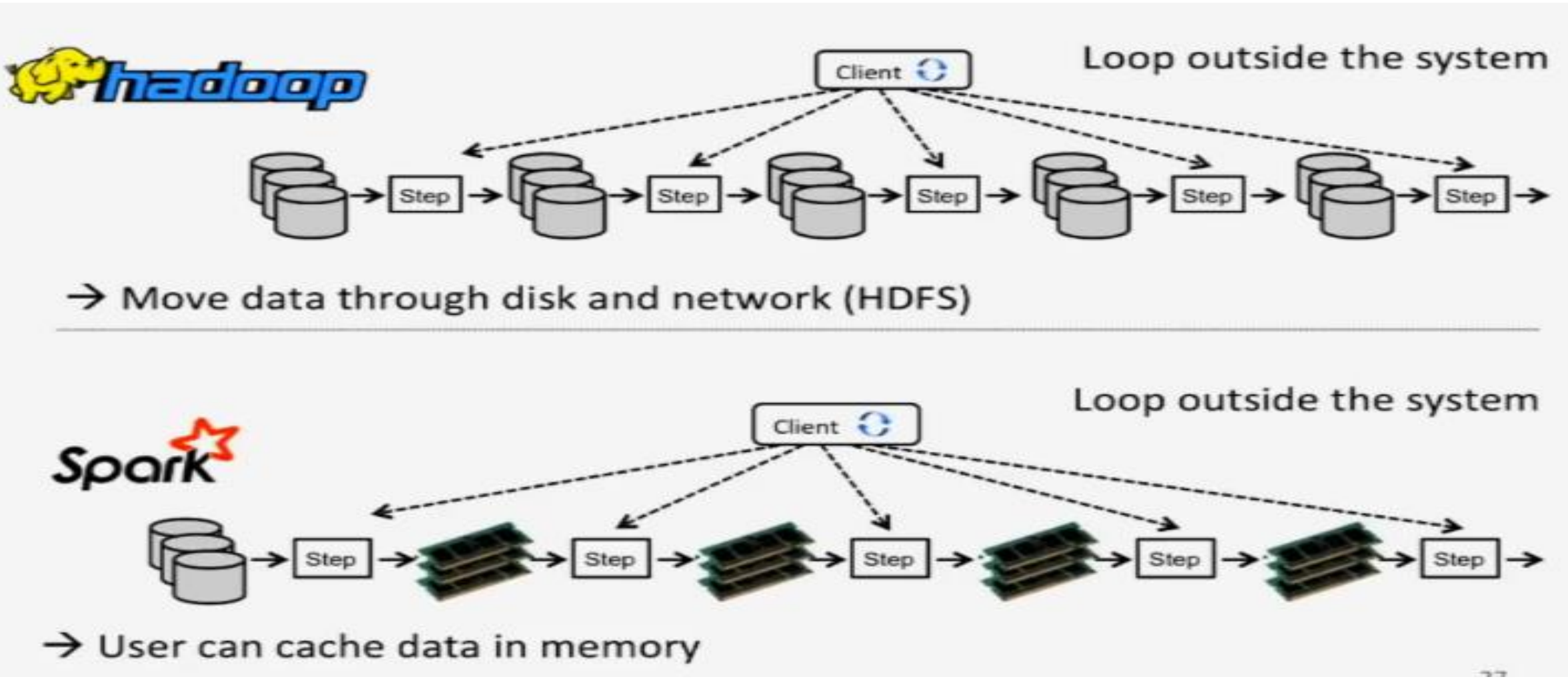
MapReduce



- Split:
  - Represent a data fragment (a data block stored by a data node)
- Map
  - Transform the split into key/value pairs
- Shuffle (optional):
  - Group back the key from each Split
- Reduce
  - Process each set of key/value pairs into a final value

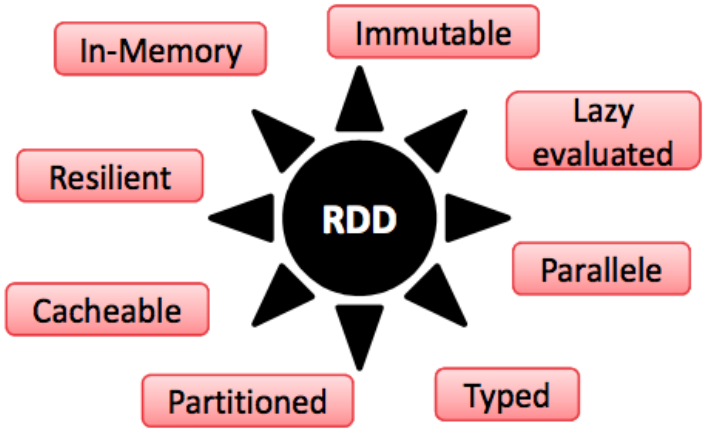


Source: https://docs.hazelcast.org/docs/3.2/manual/html/mapreduce-essentials.html


hadoop
Client
Loop outside the system
Step
Step
Step
Step
Step
→ Move data through disk and network (HDFS)
Spark
Client
Loop outside the system
Step
Step
Step
Step
Step
→ User can cache data in memory

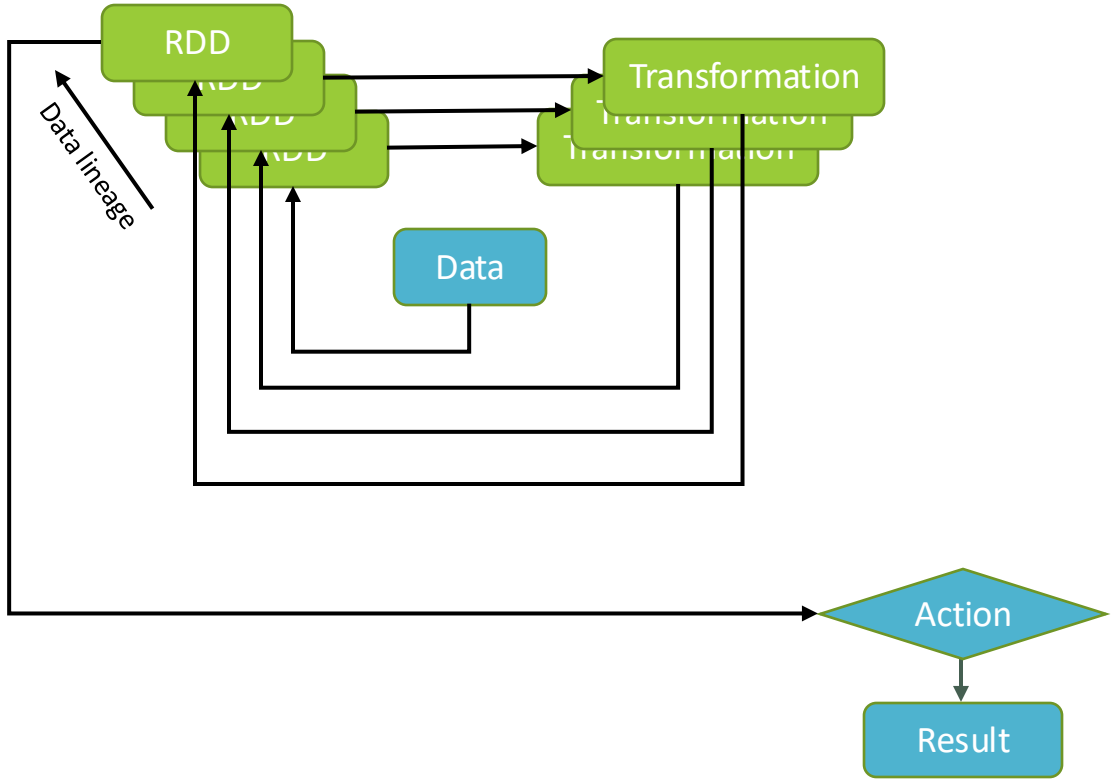
Spark Core : RDD (resilient distributed dataset)

- Immutable collection of fault tolerant elements that can be operated on "in parallel" that represents:
  - A list of partitions
  - A function for computing each split
  - A list of dependencies on other RDDs
  - An optional partitioner
  - An optional list of preferred block locations for an HDFS file



Spark Core : RDD in a nutshell

- RDD
  - immutable, iter-able, partion-able & lazy loaded data structure
  - Provide data lineage capabilities (can be reconstructed)
  - Represent each and every step of the execution
- Transformation
  - Create a new RDD from an existing one applying an operation (map, filter, Sample, Union…)
- Action:
  - Evaluate the chain of transformation on the RDD object and return a result



Spark - Transformations

- **map**(func)
- **filter**(func)
- **flatMap**(func)
- **mapPartitions**(func)
- **mapPartitionsWithIndex**(func)
- **union**(otherDataset)
- **intersection**(otherDataset)
- **distinct**([numTasks]))
- **groupByKey**([numTasks])
- **sortByKey**([ascending], [numTasks])
- **reduceByKey**(func, [numTasks])
- **aggregateByKey**(zeroValue)(seqOp, combOp, [numTasks])
- **join**(otherDataset, [numTasks])
- **cogroup**(otherDataset, [numTasks])
- **cartesian**(otherDataset)
- **pipe**(command, [envVars])
- **coalesce**(numPartitions)
- **sample**(withReplacement,fraction, seed)
- **repartition**(numPartitions)

Full List here: https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations

Spark - Actions

- **reduce**(func)
- **collect**()
- **count**()
- **first**()
- **countByKey**()
- **foreach**(func)
- **take**(n)
- **takeSample**(withReplacement,num, [seed])
- **takeOrdered**(n, [ordering])
- **saveAsTextFile**(path)
- **saveAsSequenceFile**(path) (Only Java and Scala)
- **saveAsObjectFile**(path) (Only Java and Scala)

Full List here: https://spark.apache.org/docs/latest/rdd-programming-guide.html#actions

RDD vs DataFrame vs SQL

Spark SQL:

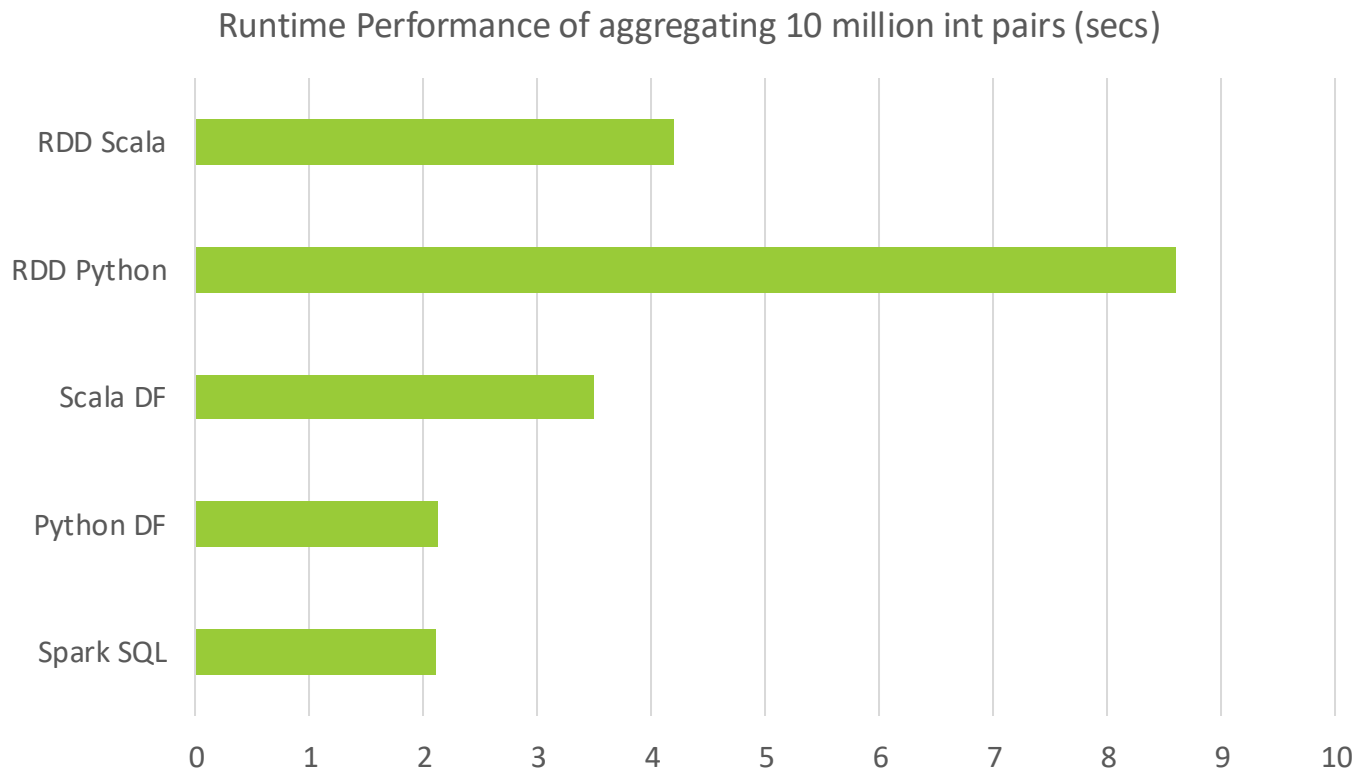
```
select dept, avg(age) from data group by dept
```

DataFrame:

```
data.groupBy("dept").avg(“age")
```

Scala RDD:

```
data.map {
  case (dept, age) => dpt -> (age, 1)
}.reduceByKey {
  case ((a1, c1), (a2, c2)) => ( a1 +a2, c1 + c2)
}.map {
  case (dept, (age, c)) => dpt -> age / c
}
```



More info: https://databricks.com/blog/2015/02/17/introducing-dataframes-in-spark-for-large-scale-data-science.html

Data Mesh

For the exam, you will have a question like (not necessarily exactly that):

- Define the concept of a “Data Mesh”, including the problem it tries to solve.
- Describe the actors and roles involved.
- Describe the pros and cons
- Describe the adoption challenges and opportunities