Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

DeepLearning.AI

Data Modeling, Transformation and Serving

Data Transformations

DeepLearning.AI

Data Transformations

Week 3 Overview

DeepLearning.AI

Batch Transformations

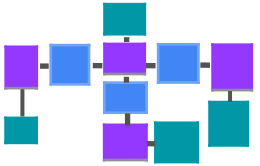
Batch Transformation Patterns and Use Cases

Transformations for Data Modeling

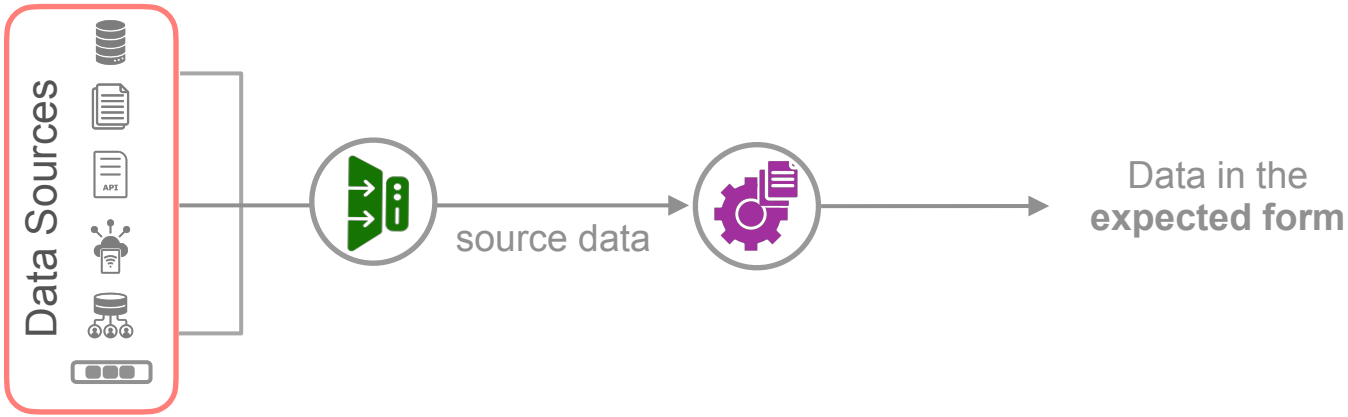
Target Model

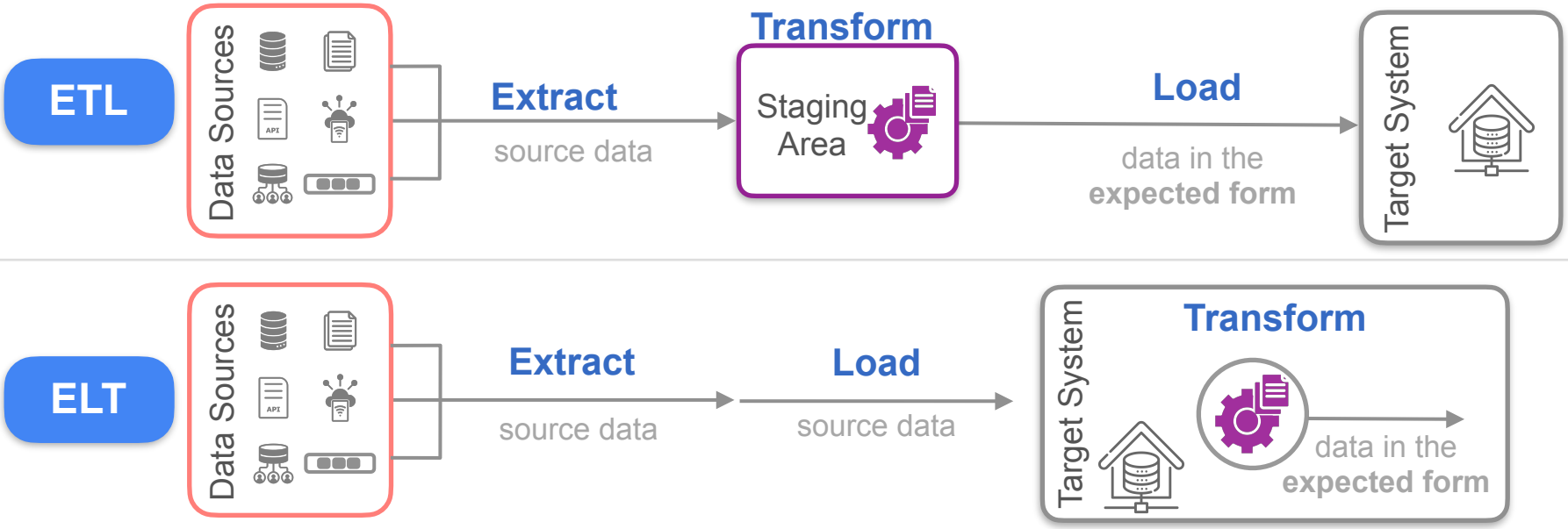
Star schemas

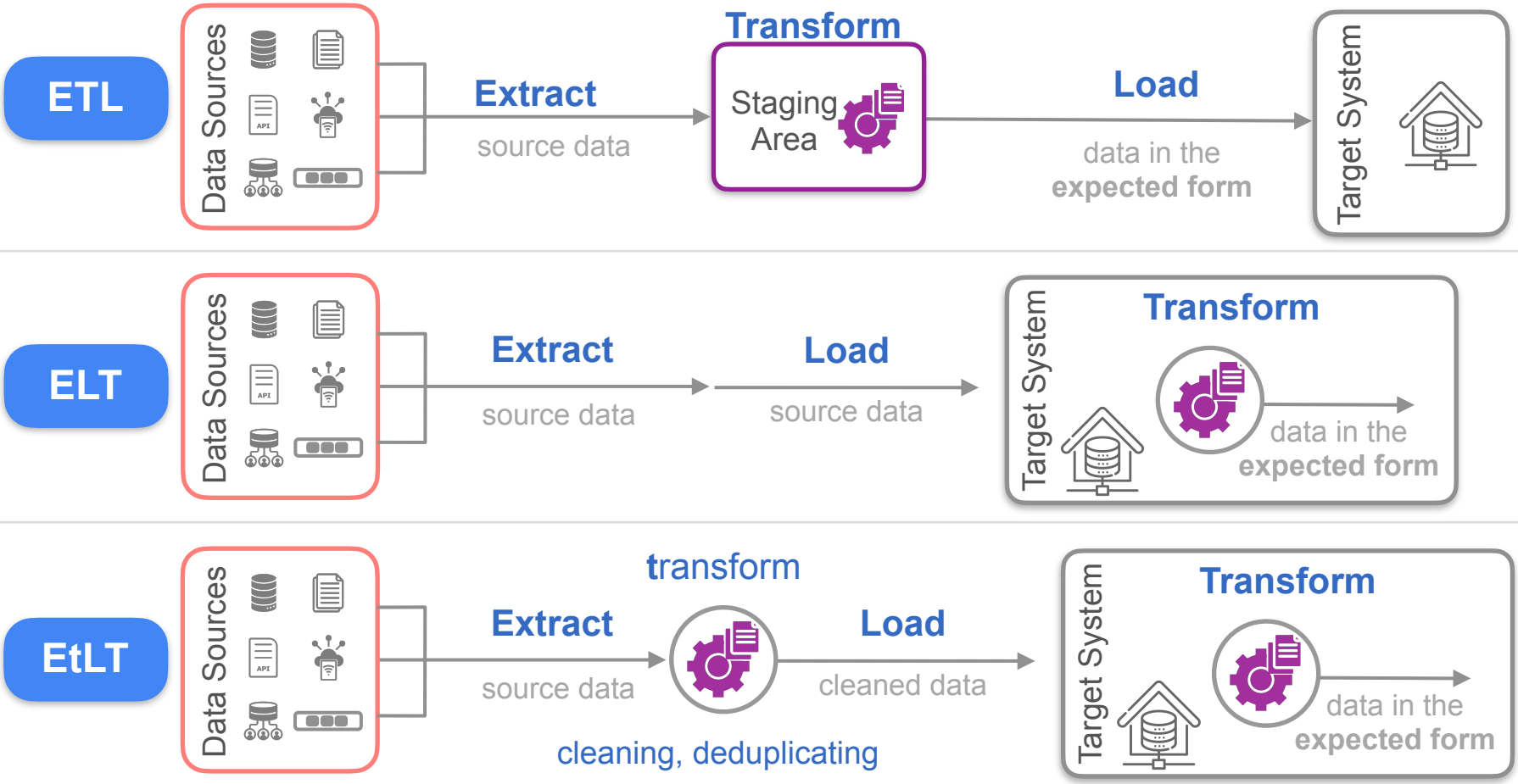
Data Vault



Restructure the source data into the **expected form**



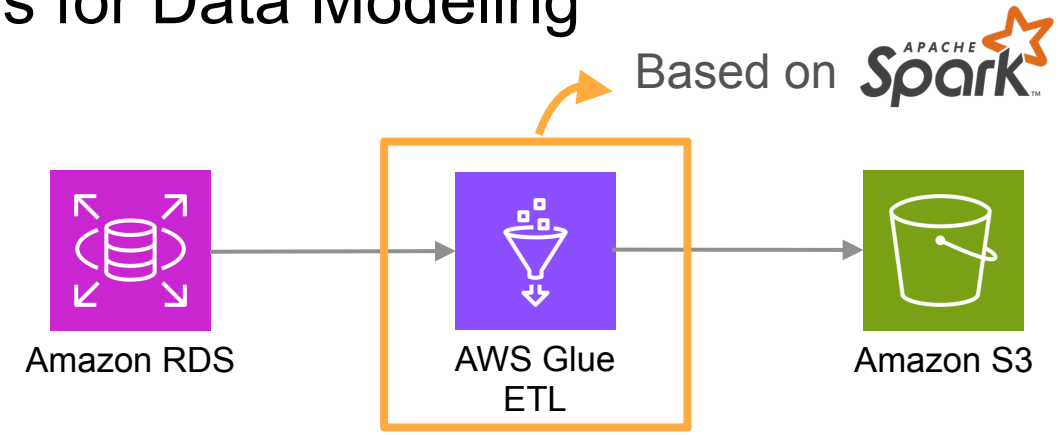

ETL
Data Sources
API
Extract
source data
Transform
Staging Area
Load
data in the expected form
Target System
ELT
Data Sources
API
Extract
source data
Load
source data
Target System
Transform
data in the expected form


ETL
Data Sources
Extract
source data
Transform
Staging Area
Load
data in the expected form
Target System
ELT
Data Sources
Extract
source data
Load
source data
Target System
Transform
data in the expected form
EtLT
Data Sources
transform
Extract
source data
Load
cleaned data
cleaning, deduplicating
Target System
Transform
data in the expected form

Transformations for Data Modeling

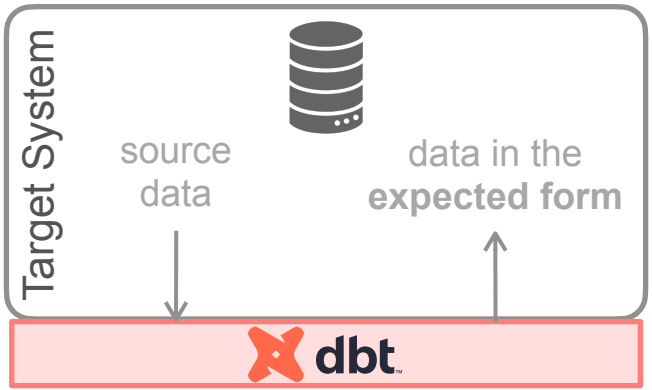
ETL

Course 1



ELT

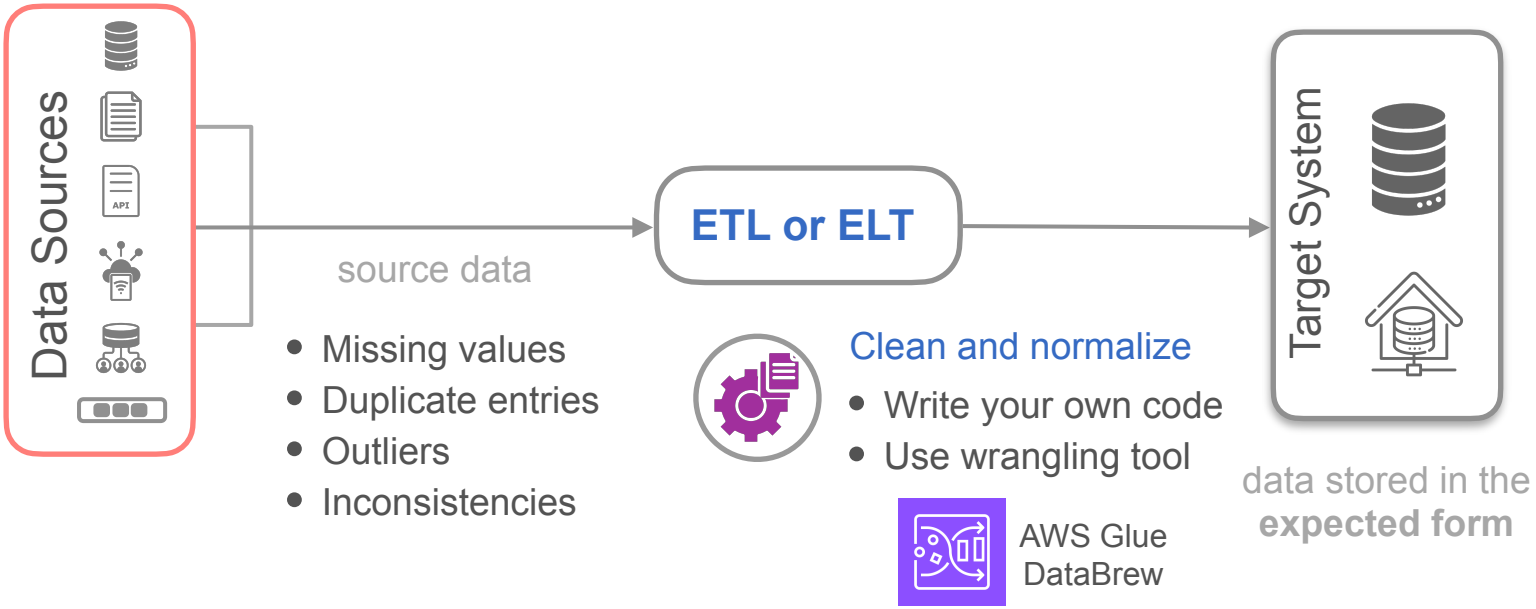
Course 4



- dbt is not an execution tool like Spark
- SQL-tool that facilitates transformation within the database or data warehouse

Transformations for Data Cleaning

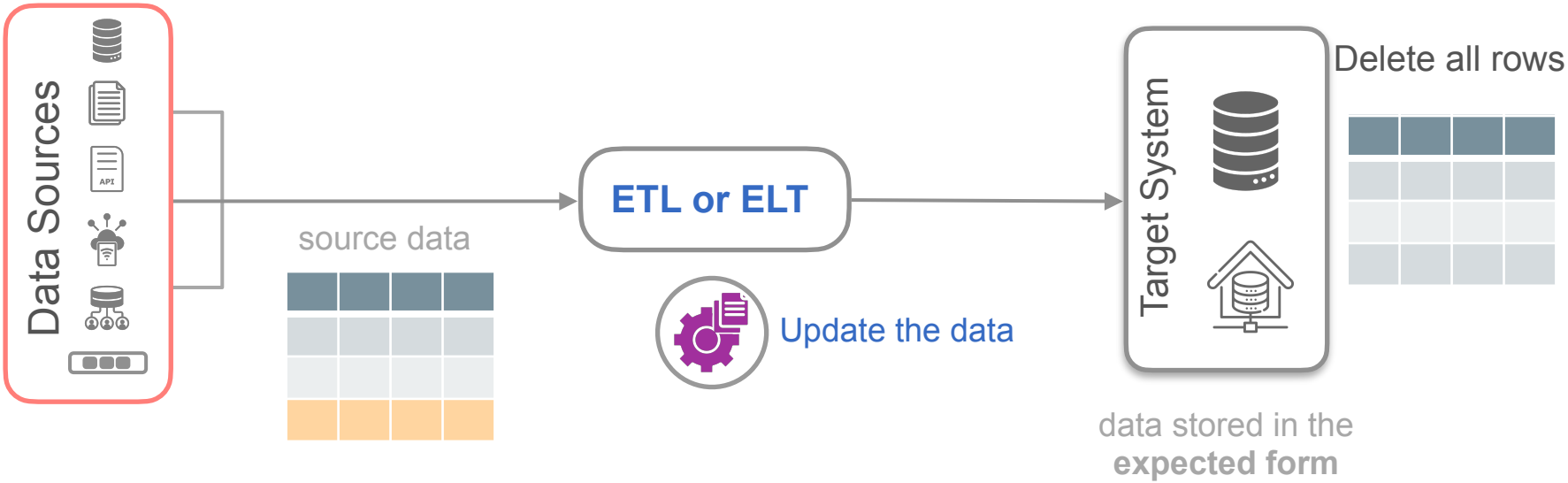
Data Wrangling Take messy, malformed data and turn it into clean data



Transformations for Data Updating

Truncate and Reload

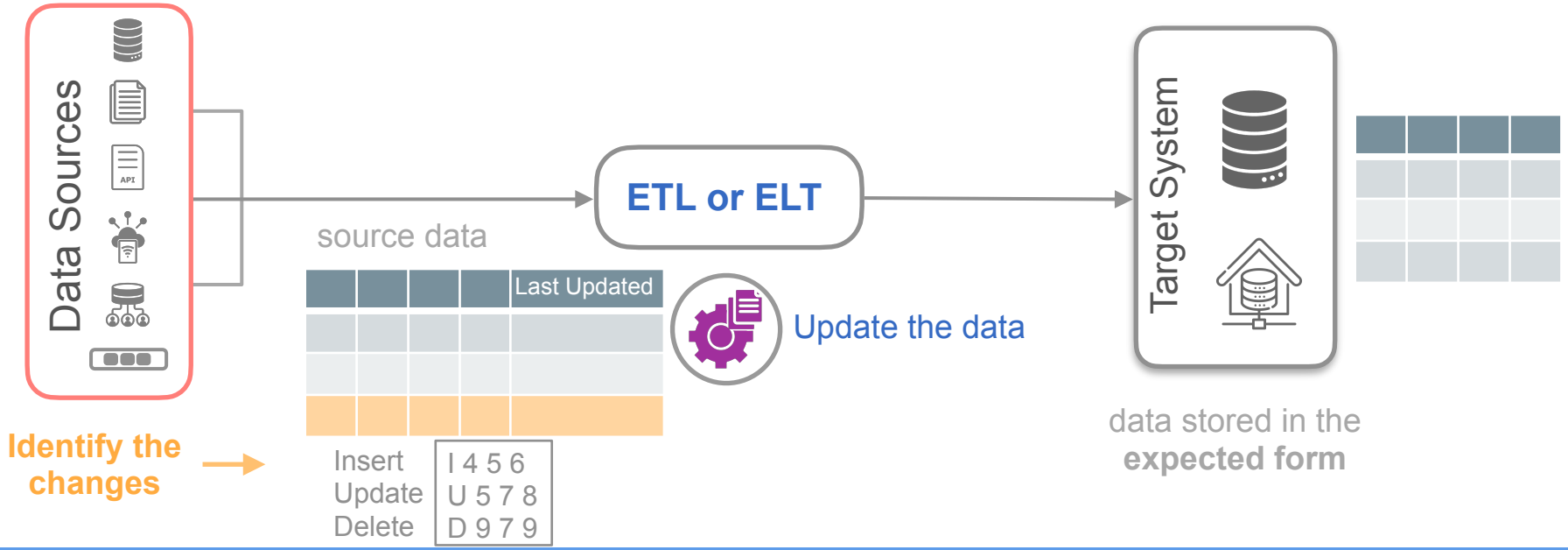
- Delete all records in target system and reload the data from the source
- Have a small dataset
- Update the data once in a while



Transformations for Data Updating

Truncate and Reload

Change Data Capture (CDC)



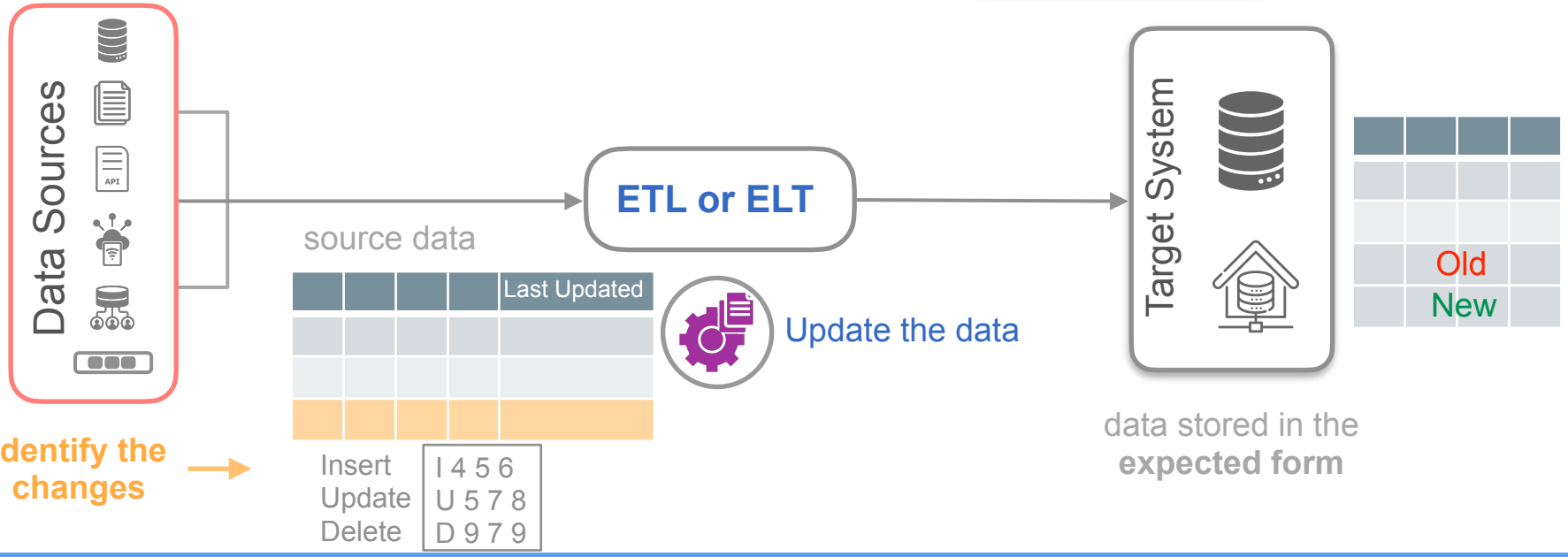
Transformations for Data Updating

Truncate and Reload

Change Data Capture (CDC)

Capture updates

- **Insert-only**
- **Upsert / Merge**



Transformations for Data Updating

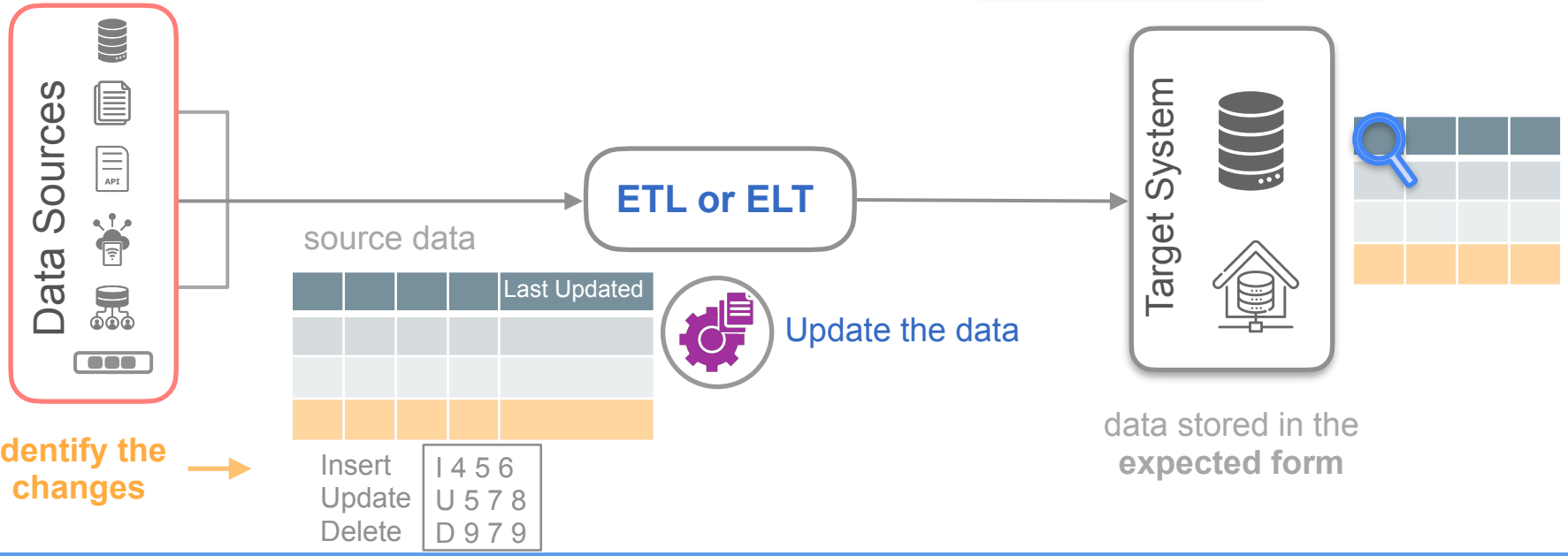
Truncate and Reload

Change Data Capture (CDC)

Capture updates

Insert-only

Upsert / Merge



Transformations for Data Updating

Truncate and Reload

Change Data Capture (CDC)

Capture updates

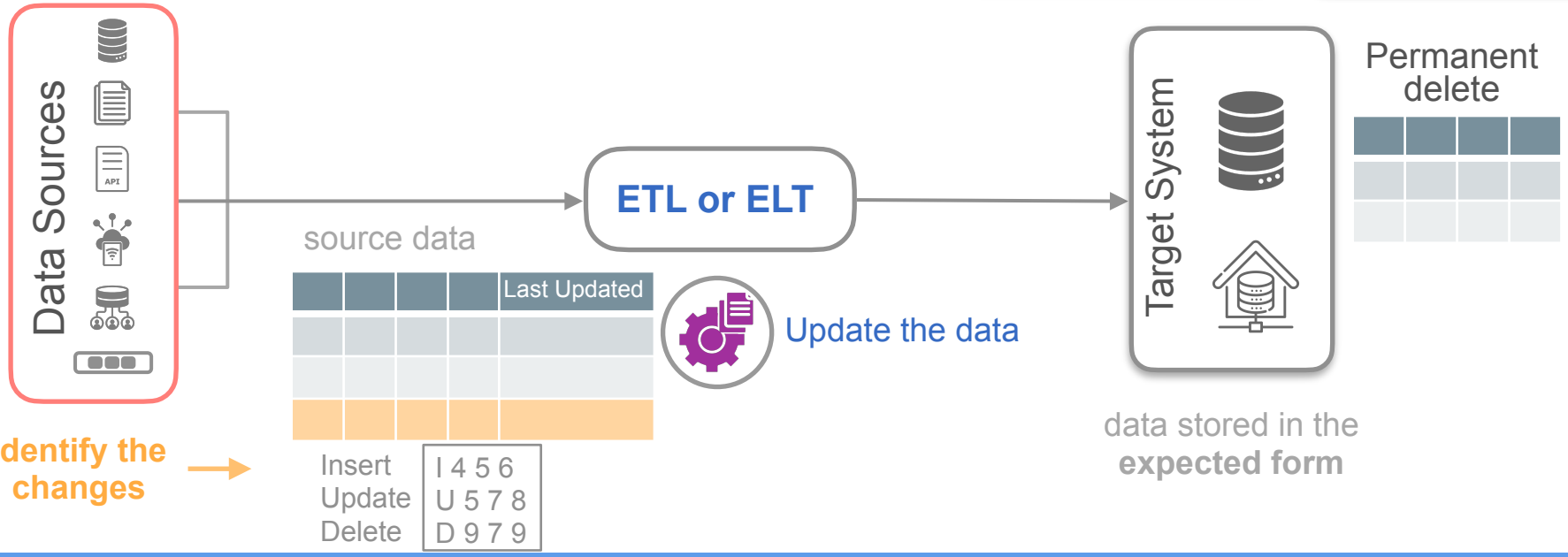
Insert-only

Upsert / Merge

Capture deletes

Hard

Soft



Transformations for Data Updating

Truncate and Reload

Change Data Capture (CDC)

Capture updates

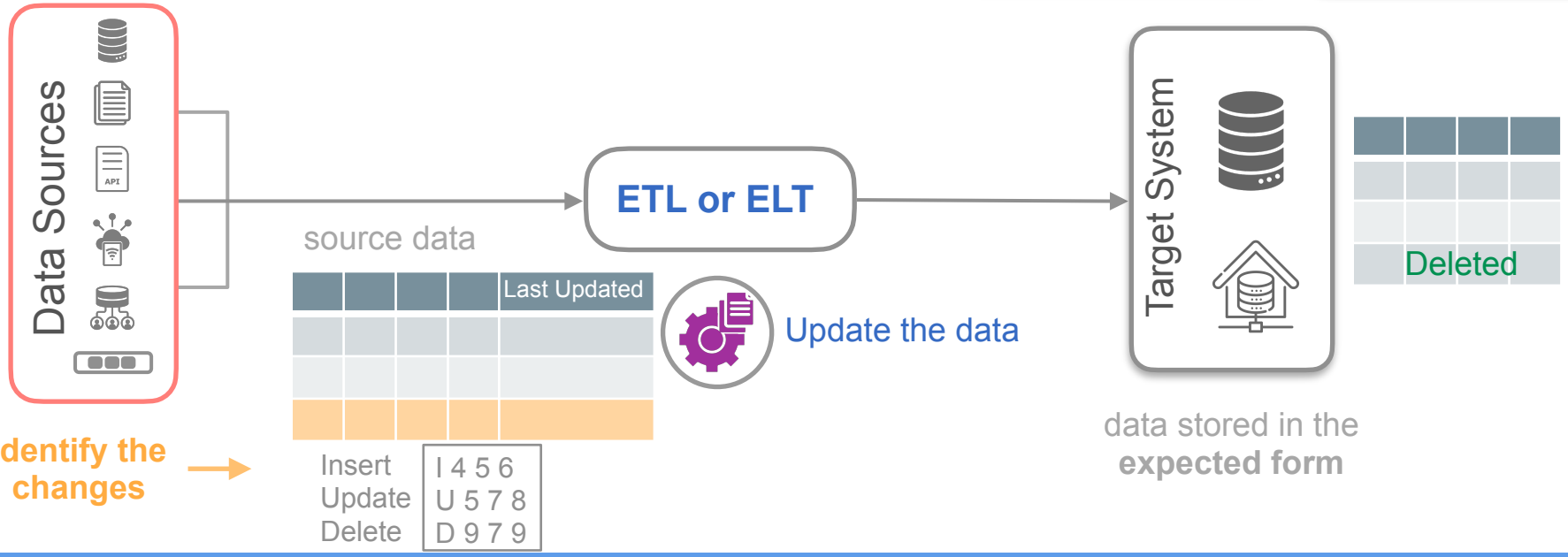
Insert-only

Upsert / Merge

Capture deletes

Hard

Soft



Transformations for Data Updating

Truncate and Reload

Change Data Capture (CDC)

Capture updates

Insert-only

Upsert / Merge

Capture deletes

Hard

Soft

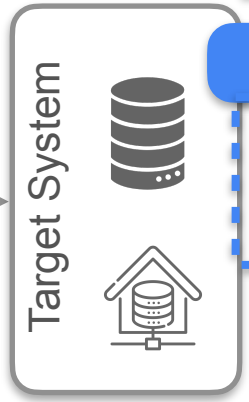
Keep all records but be able to filter them out

ETL or ELT

source data

				Last Updated

Update the data



data stored in the **expected form**

Identify the changes

Insert	I 4 5 6
Update	U 5 7 8
Delete	D 9 7 9

Transformations for Data Updating

Truncate and Reload

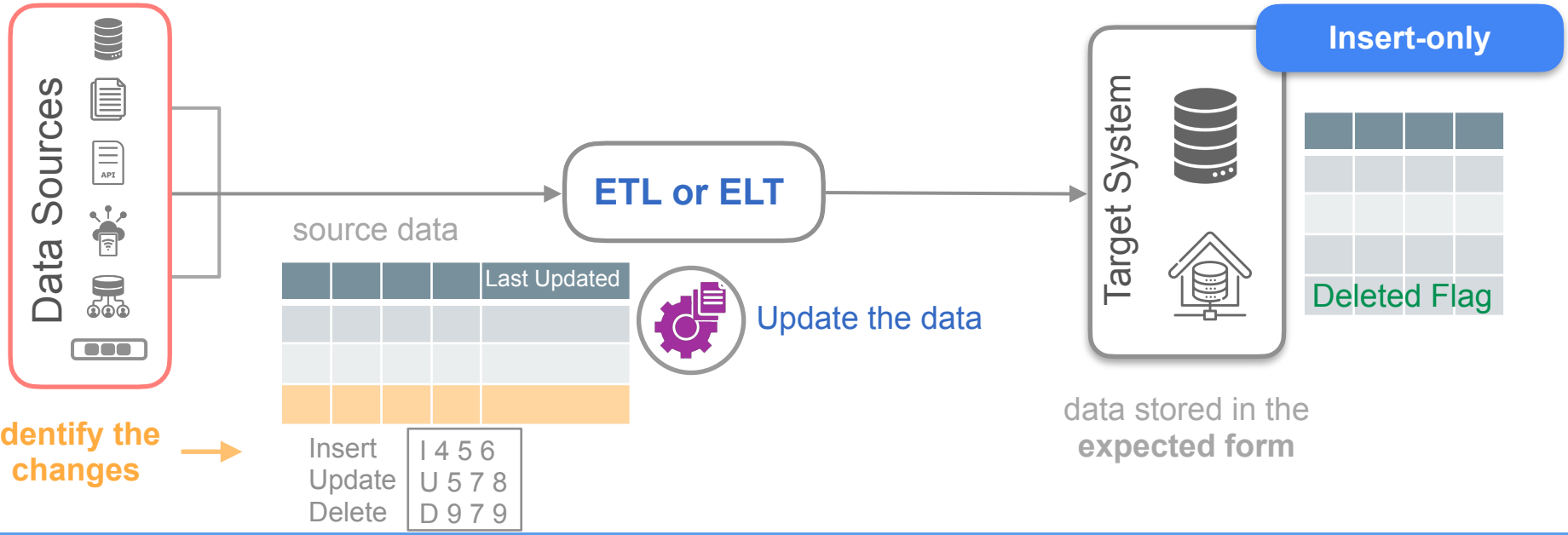
Change Data Capture (CDC)

Capture updates

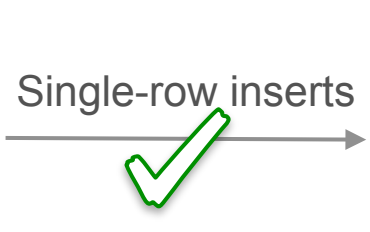
- Insert-only
- Upsert / Merge

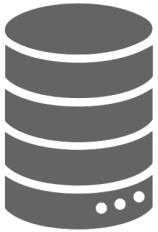
Capture deletes

- Hard
- Soft

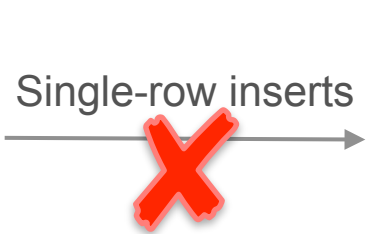


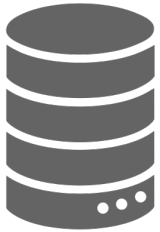
Transformations for Data Updating





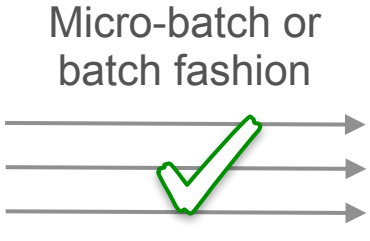
Row-oriented database

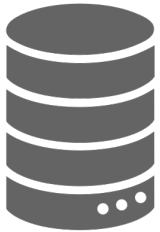




OLAP Column-oriented database

- Puts a massive load on the OLAP system
- Extremely inefficient for subsequent reads





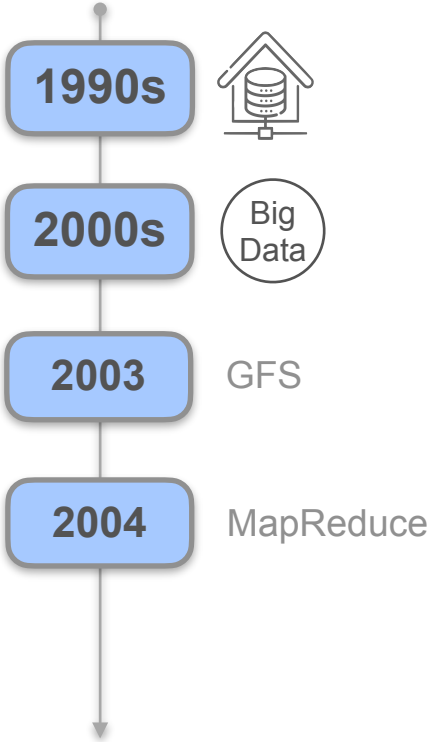
- Organizes data more efficiently
- Leverages distributed parallel processing to speed up data loading

DeepLearning.AI

Batch Transformations

Distributed Processing Framework — Hadoop MapReduce

Big Data Era



- Traditional monolithic data warehouses were not able to handle massive amounts of data effectively
- Commodity hardware became cheap and ubiquitous
- Several innovations in large-scale distributed storage and computing:

2003

Google

The Google File System

GFS

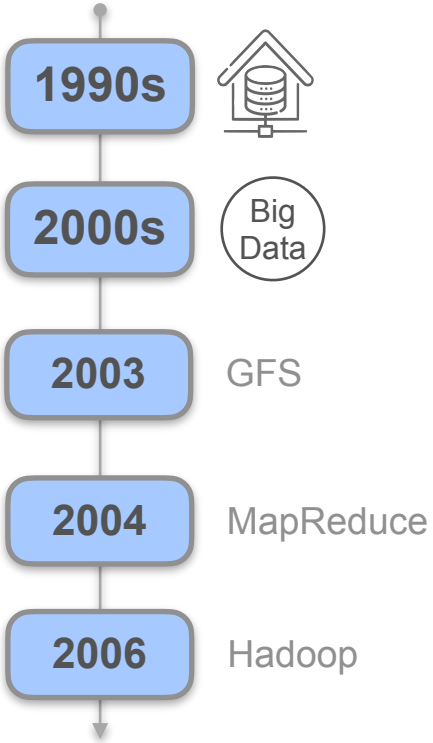
2004

Google

MapReduce

Parallel programming paradigm for processing data distributed over GFS

Big Data Era



2006

yahoo!

hadoop

- **Hadoop Distributed File System (HDFS)**
  Key ingredients of current big data engines (EMR, Spark)
- **Hadoop MapReduce**
  Influences many of today's distributed systems

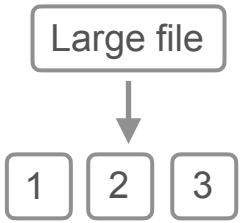
Hadoop Distributed File System

HDFS: combines compute and storage on the same nodes

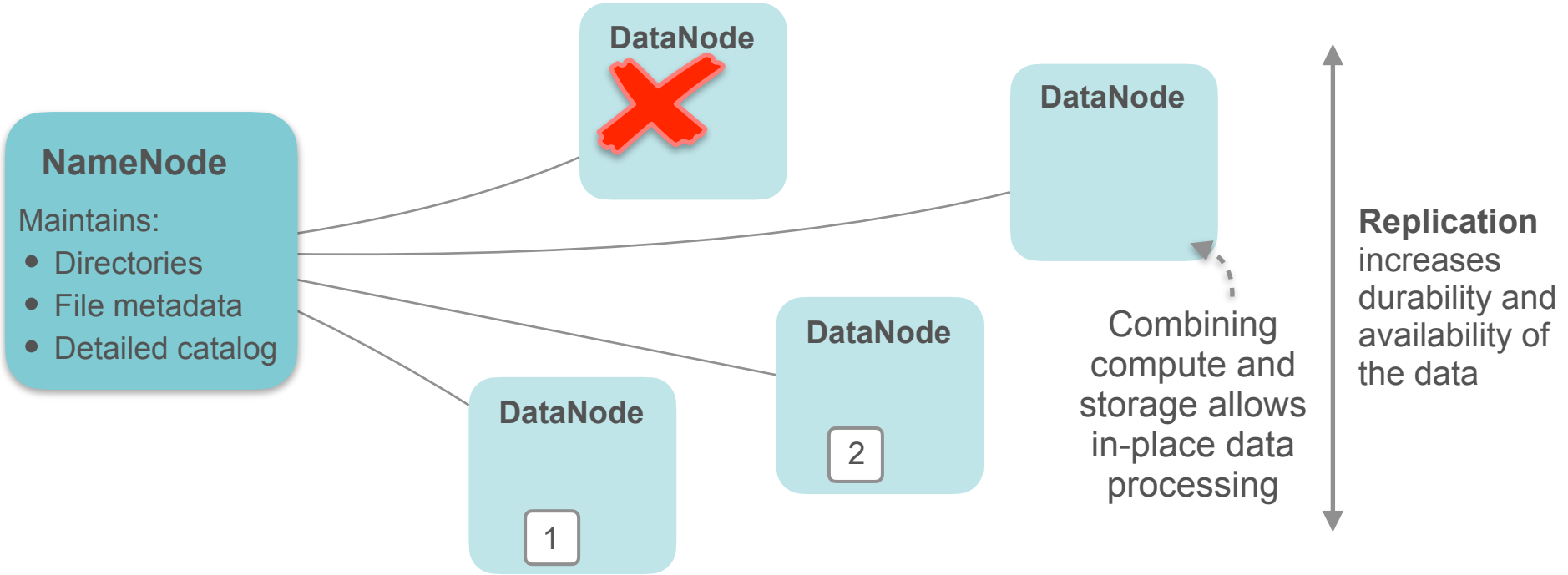
Object Storage: limited compute support for internal processing

Hadoop Distributed File System

HDFS: combines compute and storage on the same nodes

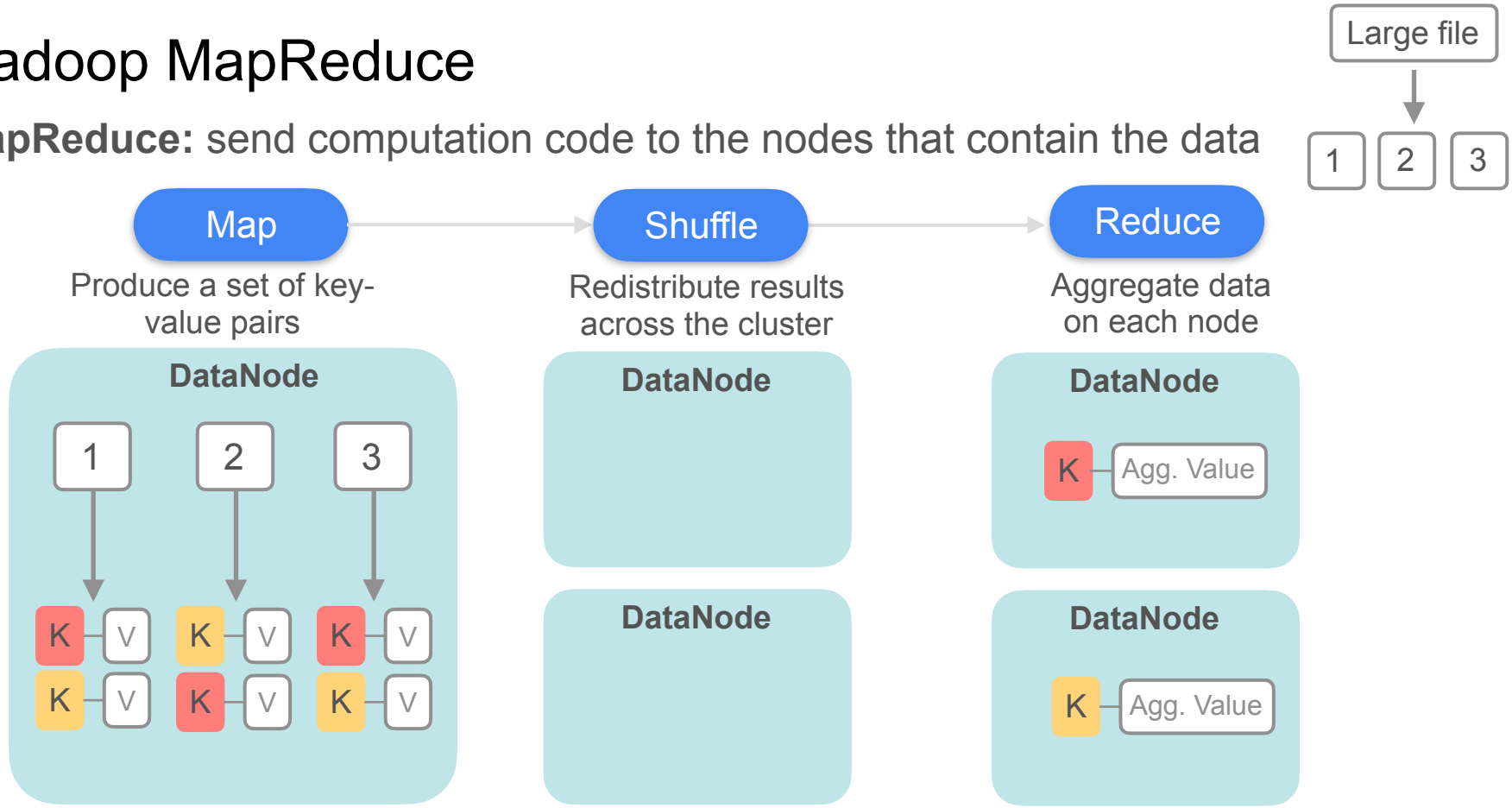


Blocks of few hundred megabytes in size



Hadoop MapReduce

MapReduce: send computation code to the nodes that contain the data



Hadoop MapReduce

```
SELECT user_id, COUNT(*)
FROM user_events
GROUP BY user_id;
```

Petabytes

user_events

1 2 3

Hundreds of Megabytes

hashing the key

Map

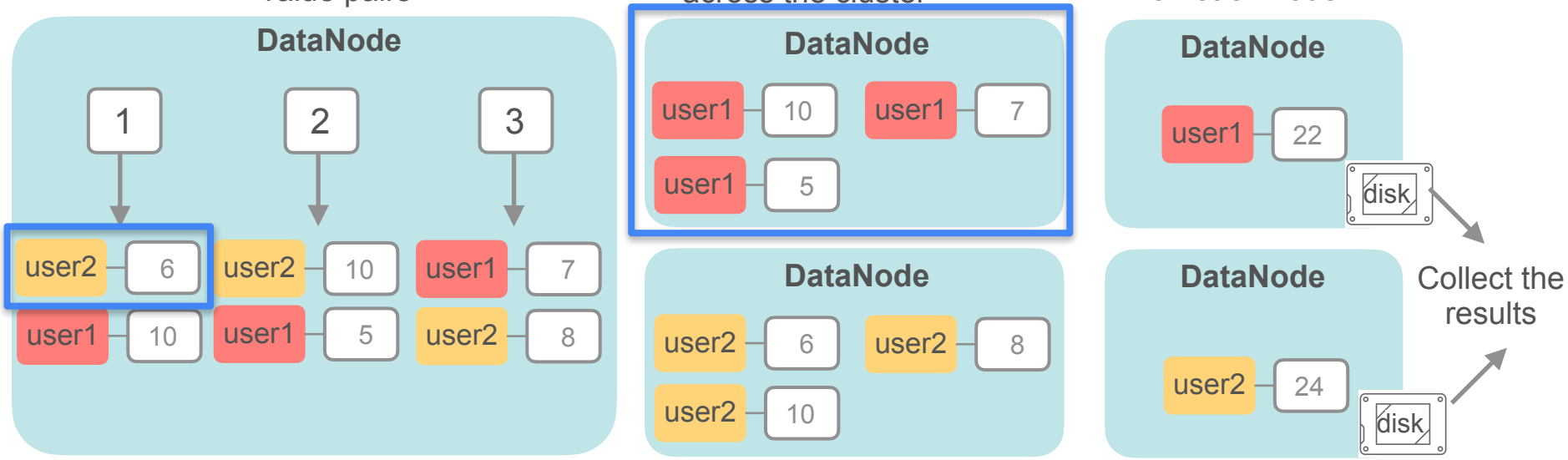
Produce a set of key-value pairs

Shuffle

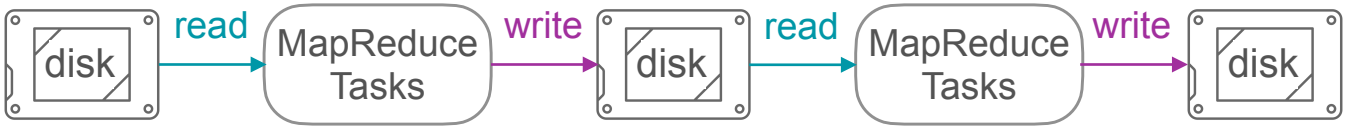
Redistribute results across the cluster

Reduce

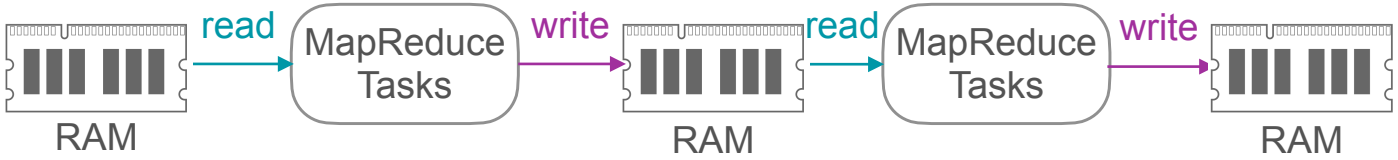
Aggregate data on each node



Hadoop MapReduce - Shortcomings



Pros	Cons
• Simplifies state and workflow management • Minimizes memory consumption	• Drives high-disk bandwidth utilization • Increases processing time



- In-memory processing speeds up data processing tasks
- Disk is treated as a second-class data storage layer

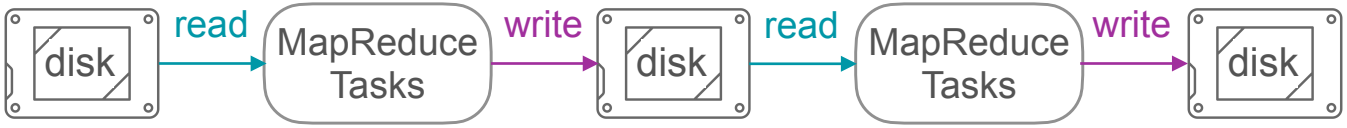
DeepLearning.AI

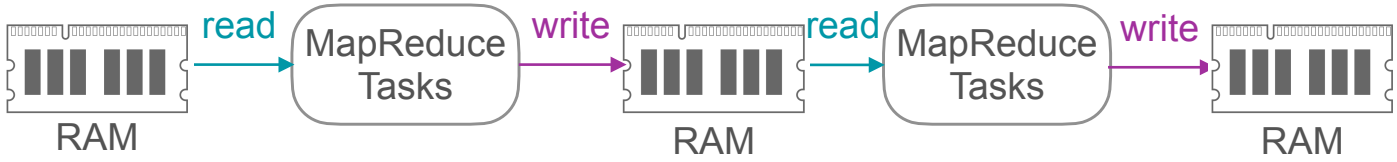
Batch Transformations

Distributed Processing Framework — Spark

Distributed processing framework

Integrate compute and persistent storage via the HDFS.





Computing engine designed for processing large datasets.

- Retains intermediate results in memory
- Limits disk I/O interactions, enabling faster computation

Separate persistent storage systems

Locally

Data centers

Cloud

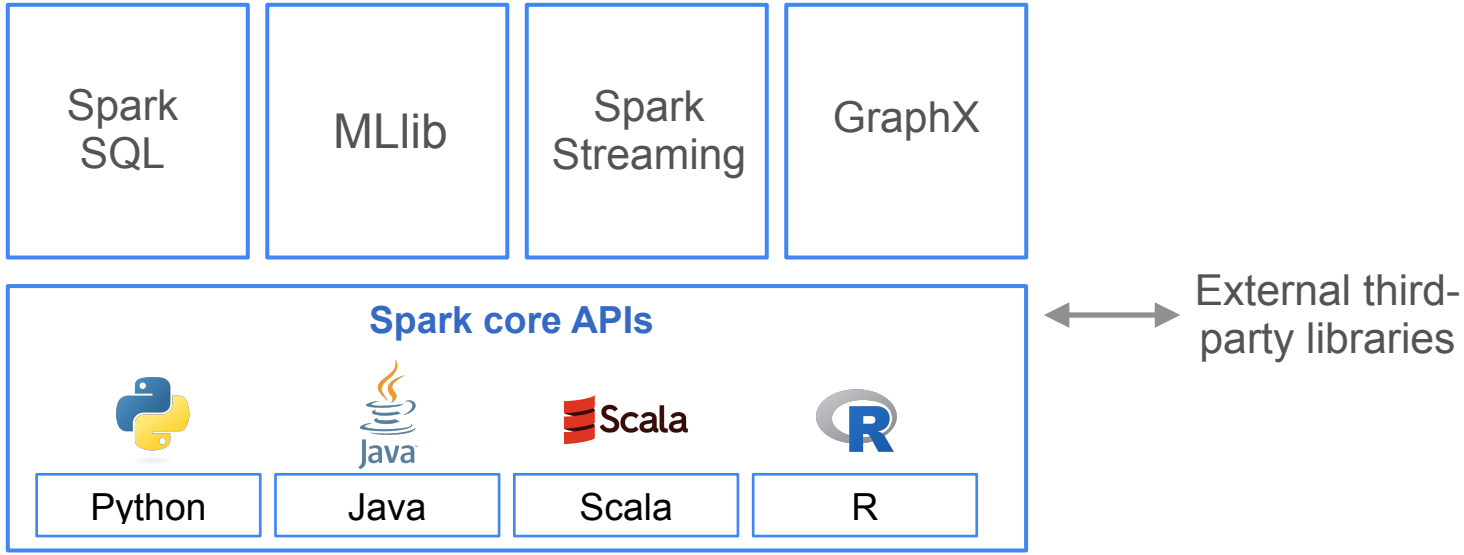
HDFS

Spark

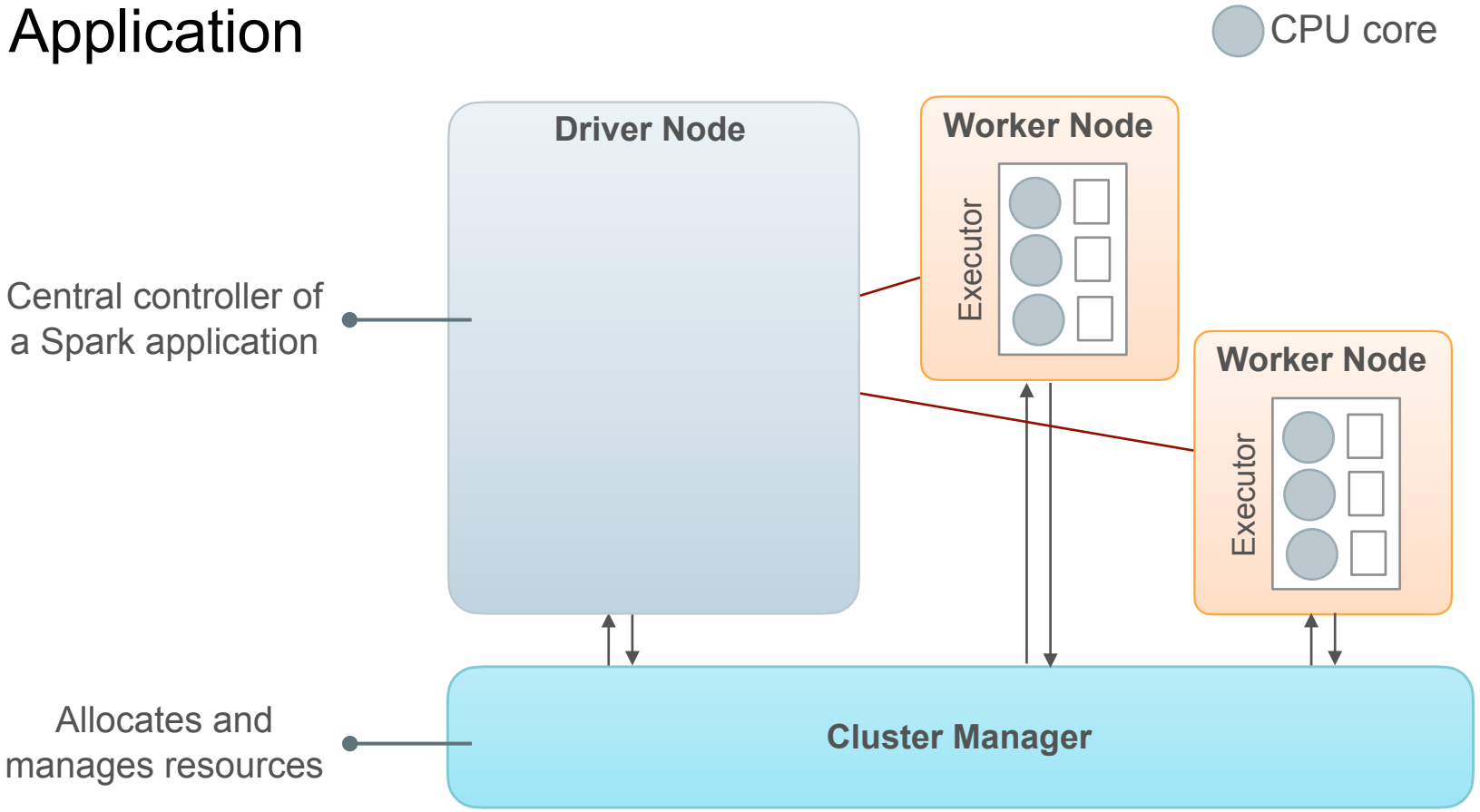
Unified Platform

You can run different types of analytical workloads:

- Perform SQL queries
- Train and test ML algorithms
- Apply streaming transformations



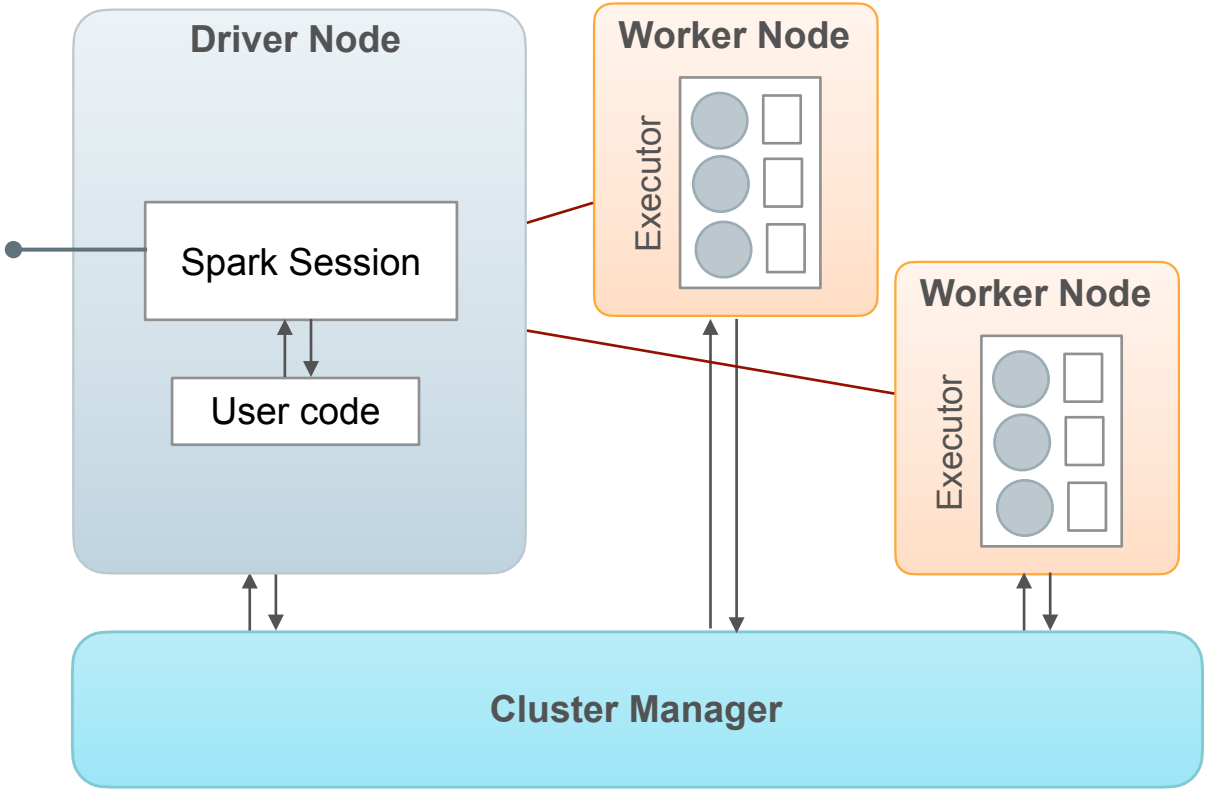
Spark Application



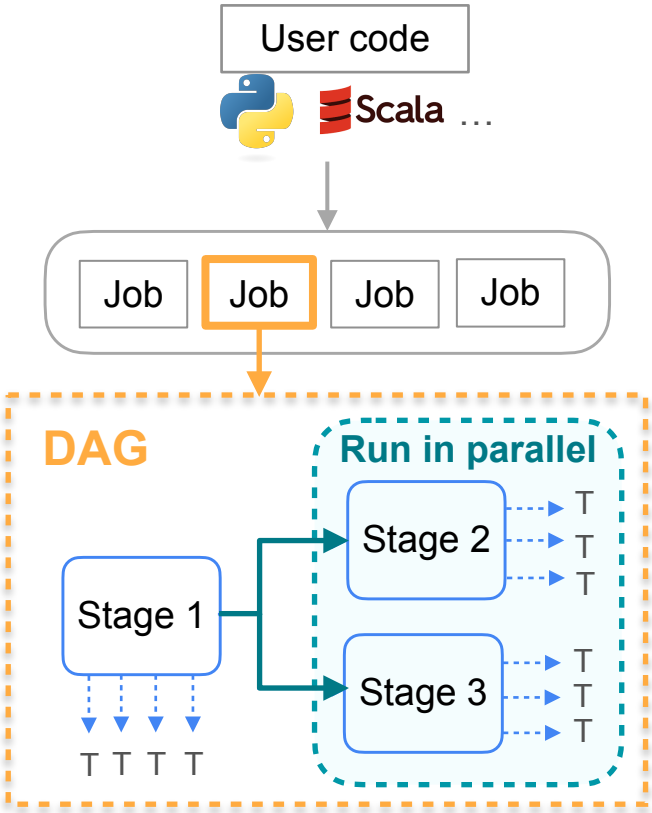
Spark Application

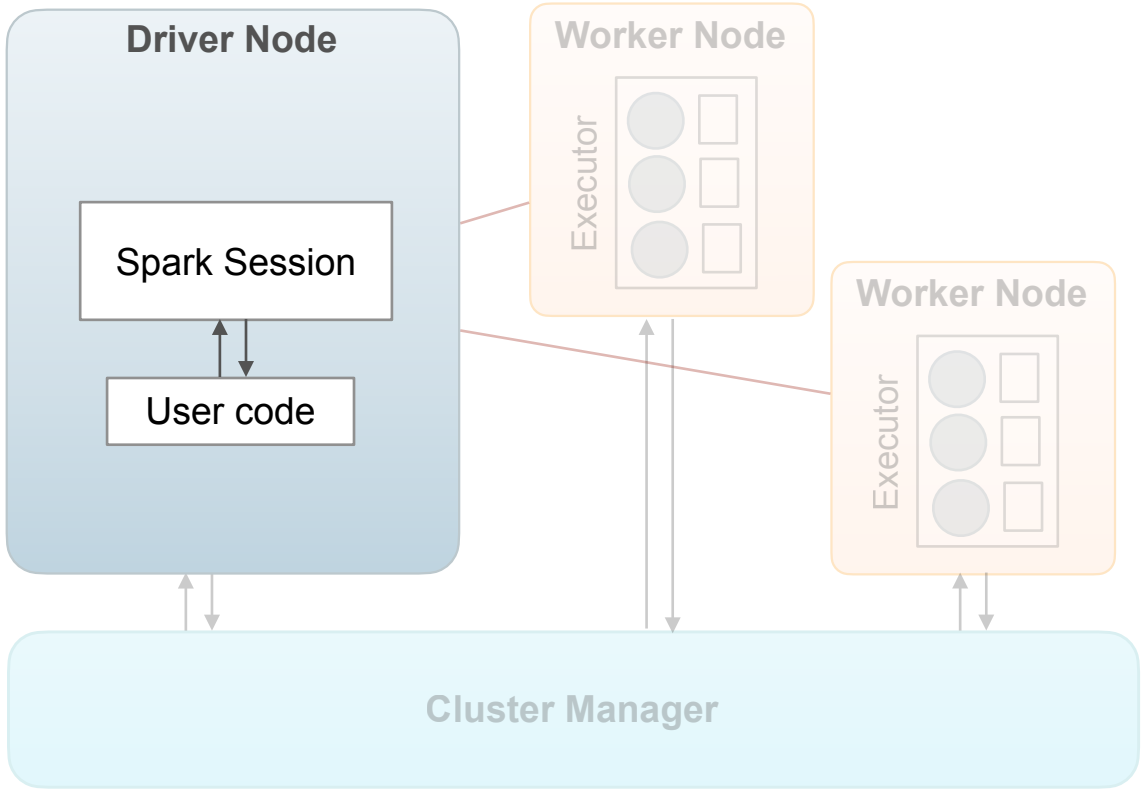
Single unified entry point to Spark's functionality:

- Define Dataframes
- Read data from sources
- Perform SQL queries

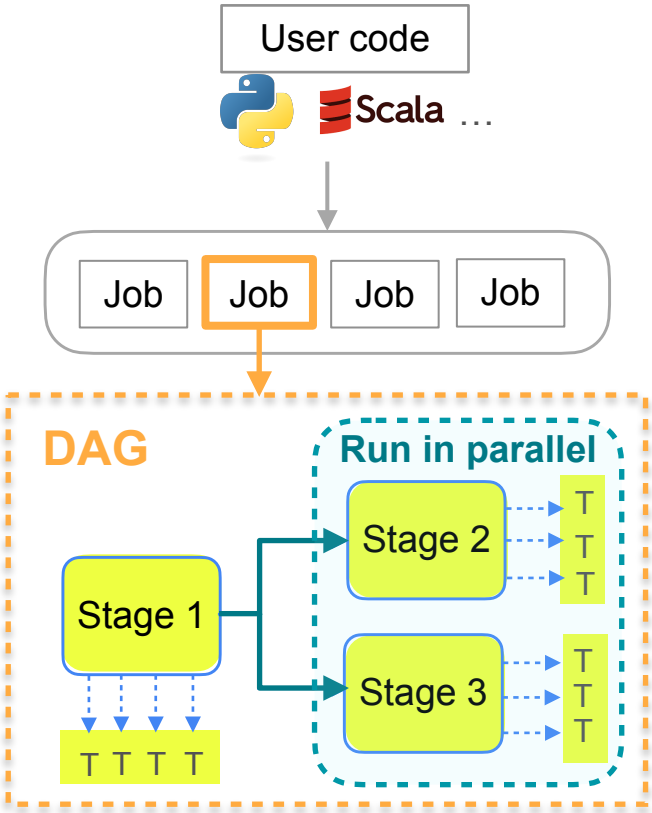


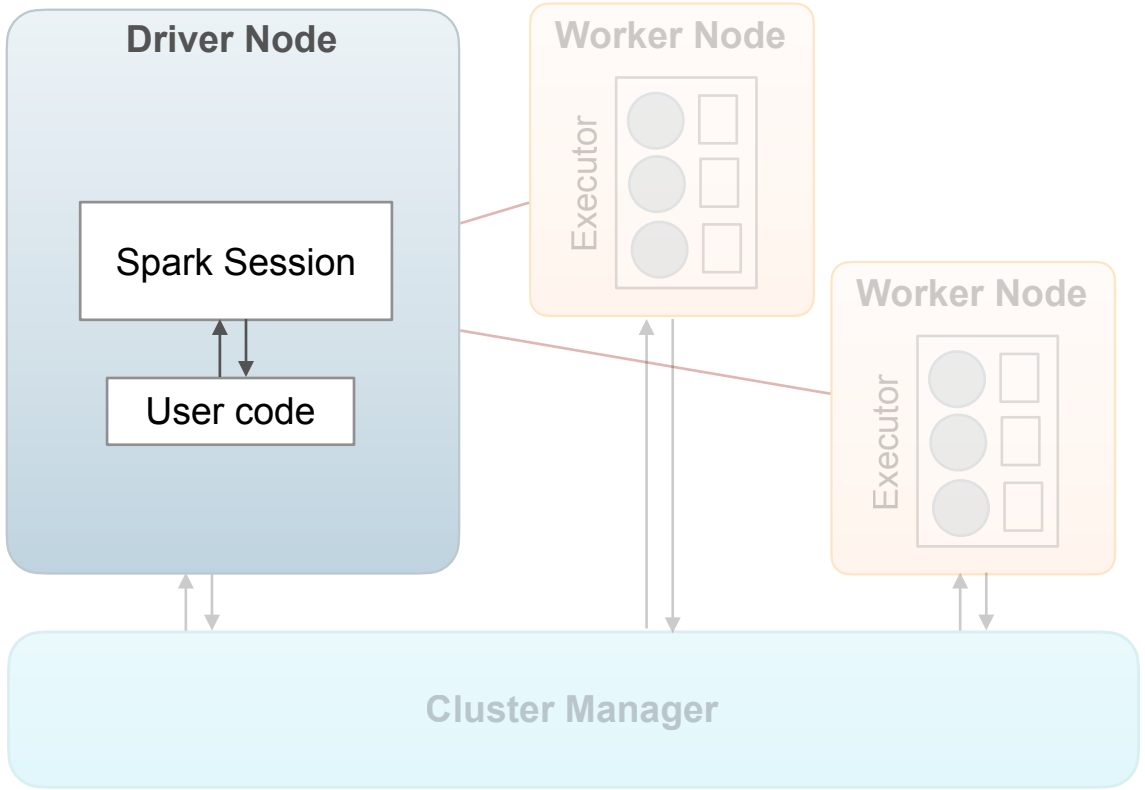
Spark Application



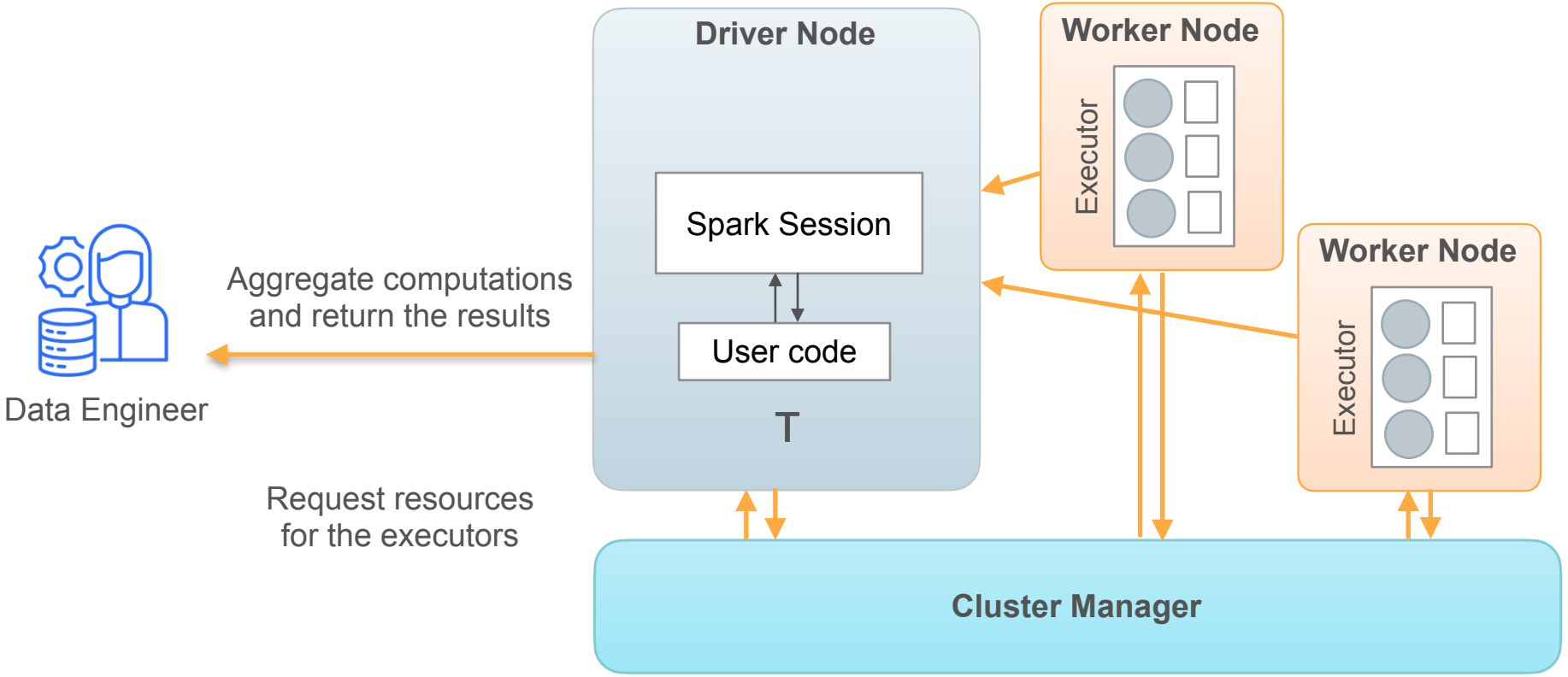


Spark Application





Spark Application



DeepLearning.AI

Batch Transformations

Spark DataFrames

Spark Dataframes

High-level Structured API

Spark DataFrames

Use simpler and more expressive operations:
Filtering, selecting, counting, aggregating and grouping

Immutable data structures

Low-level API

Resilient Distributed Dataset (RDD)

actual partitioned collection of records

Need to manually define and optimize operations

Spark Dataframes

High-level Structured API

Spark DataFrames

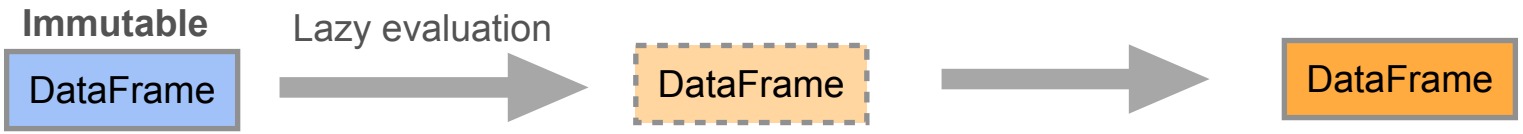
Use simpler and more expressive operations:
Filtering, selecting, counting, aggregating and grouping

Transformation

Filtering, selecting, joining and grouping

Action

Counting, showing, and saving



- Transformations: recorded as a lineage, executed when an action is invoked
- Allows Spark to optimize the execution plan
- Lineage & immutability: fault-tolerance of DataFrames

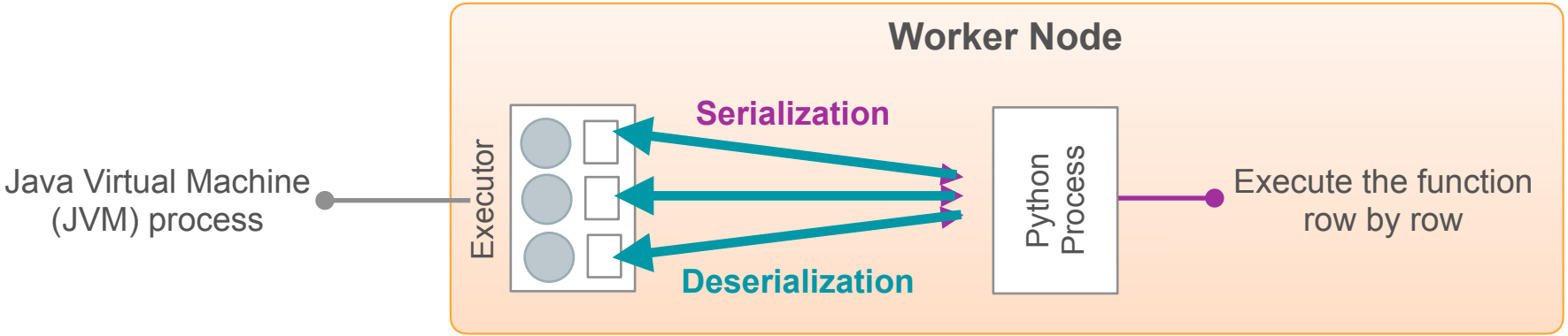
DeepLearning.AI

Batch Transformations

Demo: Working with Spark DataFrames Using Python

Python UDFs

Spark is native in Scala



- The JVM and Python processes will compete for memory resources.
- Serialization and deserialization are expensive.
- Consider writing UDFs in Scala or Java: they run directly within the JVM.

DeepLearning.AI

Batch Transformations

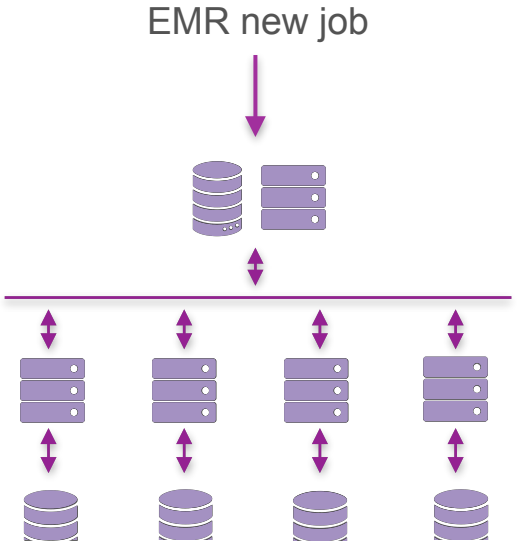
Demo: Working with Spark SQL

DeepLearning.AI

Batch Transformations

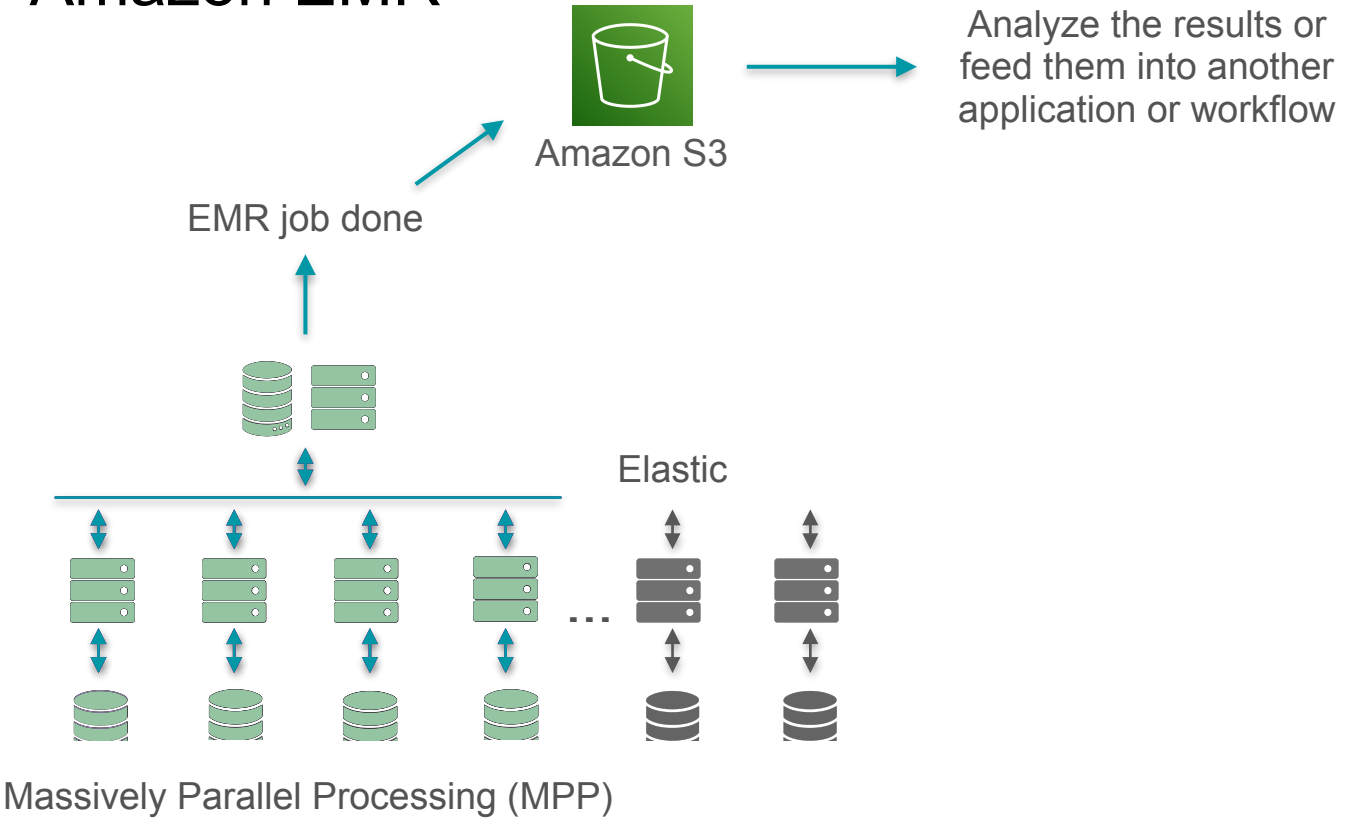
EMR

Amazon EMR



Massively Parallel Processing (MPP)

Amazon EMR


Amazon S3
Analyze the results or feed them into another application or workflow
EMR job done
Elastic
Massively Parallel Processing (MPP)

Amazon EMR

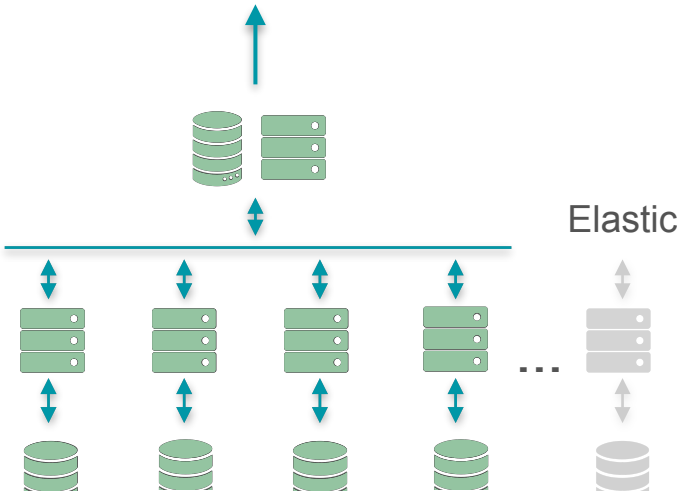
Amazon S3

Amazon EMR

Management Environment

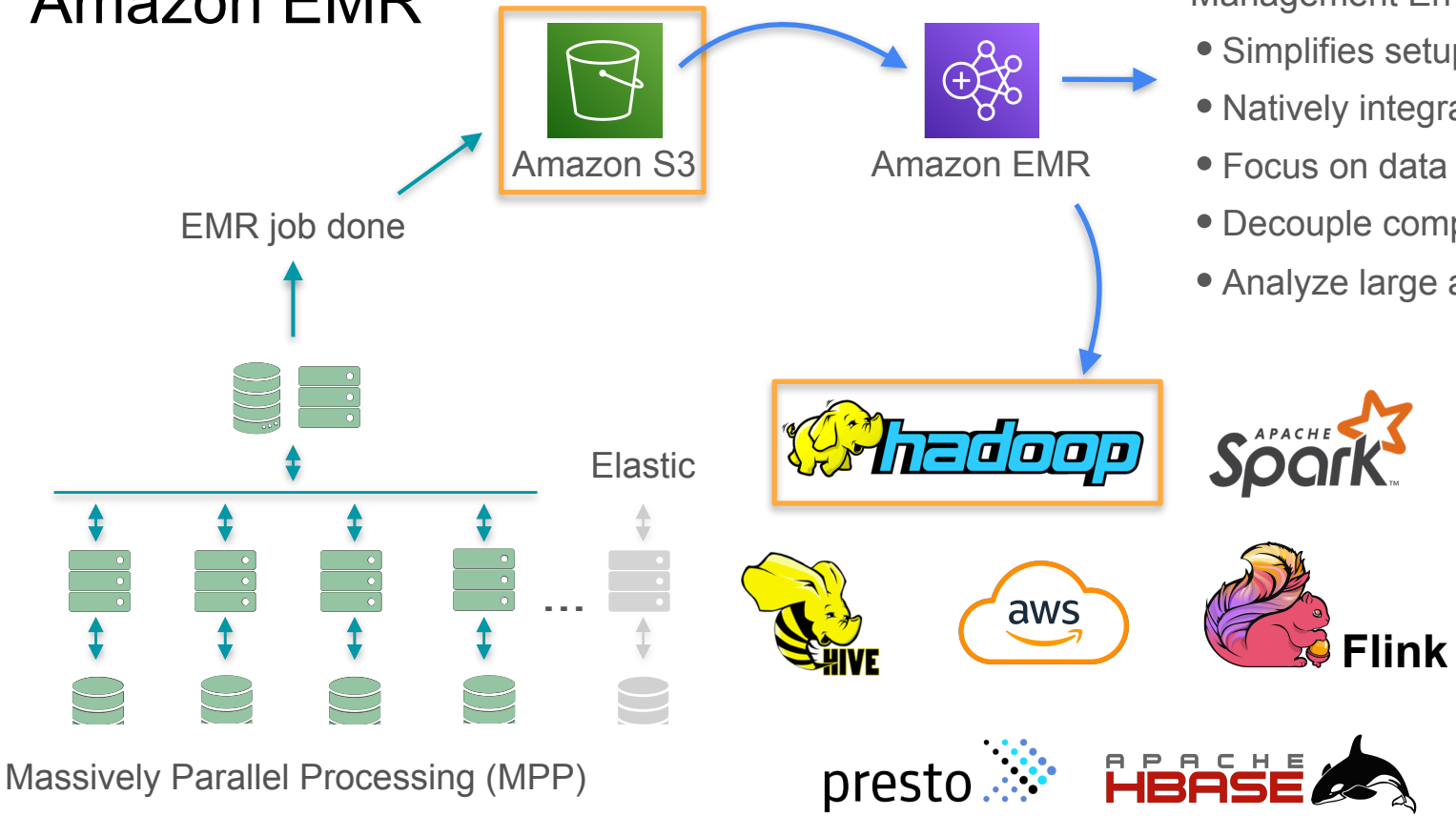
- Simplifies setup and scaling
- Natively integrates with AWS
- Focus on data workflows

EMR job done



Massively Parallel Processing (MPP)

Amazon EMR



Management Environment

- Simplifies setup and scaling
- Natively integrates with AWS
- Focus on data workflows
- Decouple compute and storage
- Analyze large amount of data

Amazon EMR

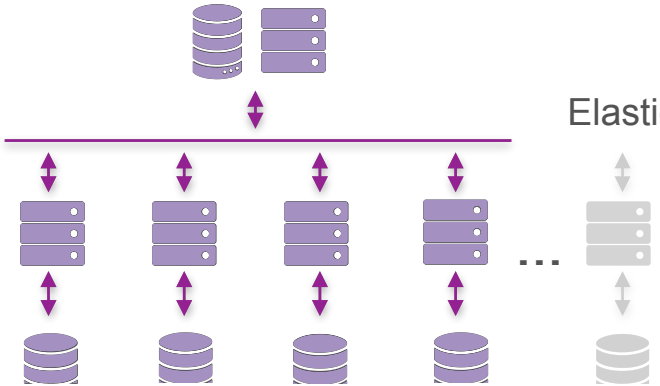
Amazon S3

Amazon EMR

Management Environment

- Simplifies setup and scaling
- Natively integrates with AWS
- Focus on data workflows
- Decouple compute and storage
- Analyze large amount of data

EMR workflow



Elastic

Massively Parallel Processing (MPP)

Amazon EMR

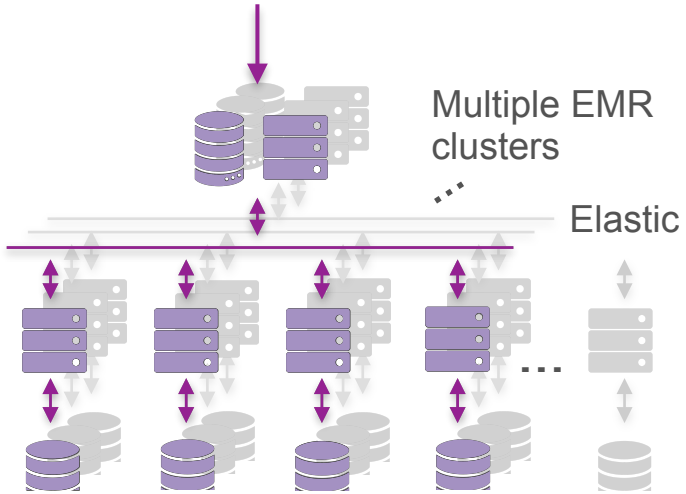
Amazon S3

Amazon EMR

Management Environment

- Simplifies setup and scaling
- Natively integrates with AWS
- Focus on data workflows
- Decouple compute and storage
- Analyze large amount of data

EMR workflow



Massively Parallel Processing (MPP)

Amazon DynamoDB

Amazon RDS

Amazon Redshift

DeepLearning.AI

Batch Transformations

Technical Considerations

Transformation Complexity	Code Reusability and Testability	Team's skills and background

Simple transformation

Python and SQL approaches have comparable performance:

- translated to the same execution plan
- executed by the same engine

Complex transformation

Cannot be implemented in SQL / not in a straightforward way

- transpose operation:
  - DataFrame: df.T
  - SQL: Not supported
- normalizing and cleaning data:
  - Python may require more code than SQL

Transformation Complexity	Code Reusability and Testability	Team's skills and background

Simple transformation
Python and SQL approaches have comparable performance:
• translated to the same execution plan
• executed by the same engine
 	
Complex transformation
Cannot be implemented in SQL / not in a straightforward way
• transpose operation:
  • DataFrame: df.T
  • SQL: Not supported
• normalizing and cleaning data:
  • Python may require more code than SQL
	• DataFrames: more testable, maintainable, modular/ reusable code.
	
	• SQL: no good notion of reusability for more complex query component	

Transformation Complexity	Code Reusability and Testability	Team's skills and background

Simple transformation
Python and SQL approaches have comparable performance:
• translated to the same execution plan
• executed by the same engine
 	
Complex transformation
Cannot be implemented in SQL / not in a straightforward way
• transpose operation:
  • DataFrame: df.T
  • SQL: Not supported
• normalizing and cleaning data:
  • Python may require more code than SQL
	• DataFrames: more testable, maintainable, modular/ reusable code.
	
	• SQL: no good notion of reusability for more complex query component
		• SQL: writing SQL queries might be simpler and easier than working with DataFrames

When to use Spark DataFrames?

Distributed Framework: load the entire data into a cluster of nodes

Use Spark: if data doesn't fit entirely into memory or you want to leverage distributed computations

Non-distributed Framework: load the entire data into the memory

Use pandas: if data can fit into the memory

Best Practices

- Extract only the data you need from the source
- Apply transformation inside the source database to reduce the size of the ingested data

DeepLearning.AI

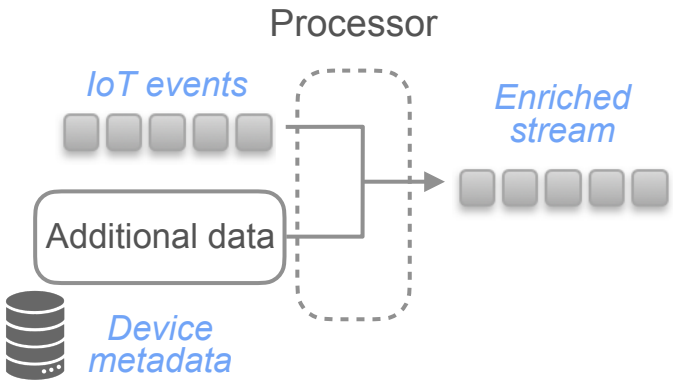
Streaming Transformations

Stream Processing

Streaming Transformations

Prepares data for downstream consumption by converting a stream of events into another stream

Enrich a Stream



Join Two Streams

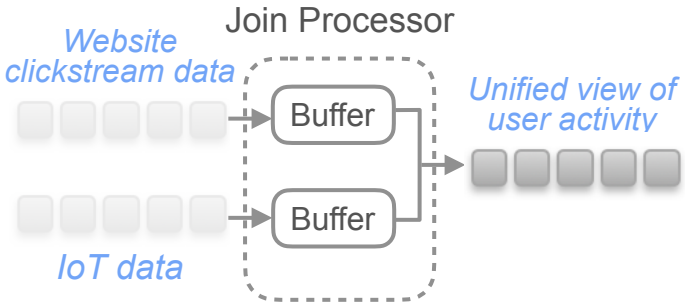
C2W2 Streaming Ingestion Lab

Streaming Transformations

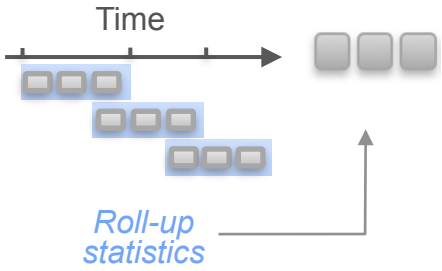
Prepares data for downstream consumption by converting a stream of events into another stream

Enrich a Stream

Join Two Streams



Windowed queries

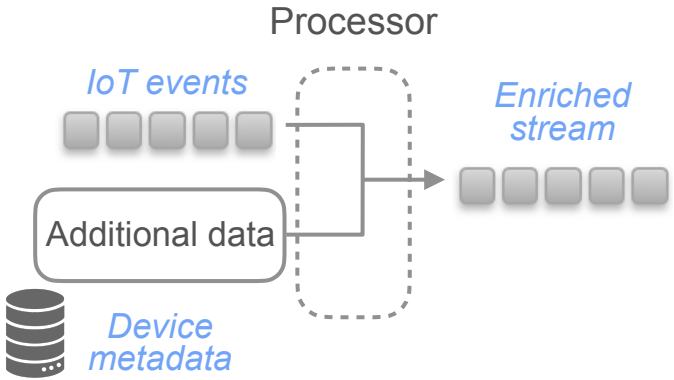


C2W2 Streaming Ingestion Lab

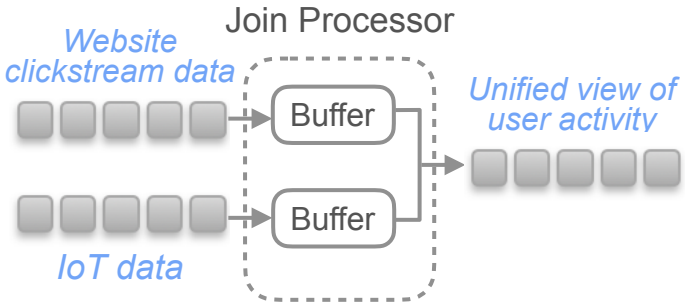
Streaming Transformations

Prepares data for downstream consumption by converting a stream of events into another stream

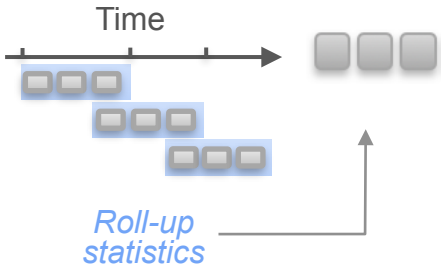
Enrich a Stream

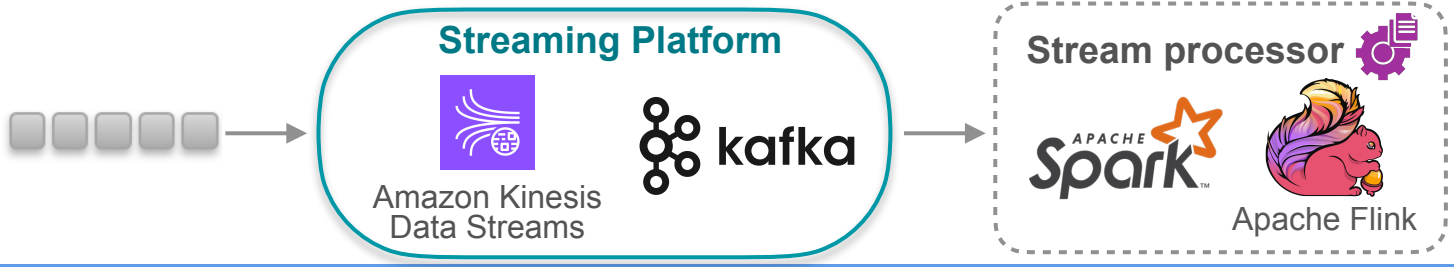


Join Two Streams



Windowed queries





- Distributed stream processing tools
- Open-source
- Python or SQL queries

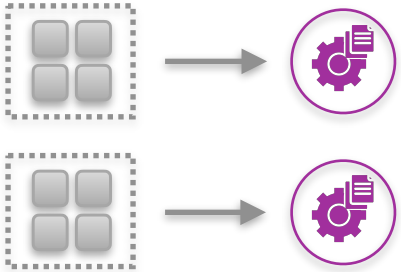
Considerations for Choosing a Stream Processing Tool

- Your use case
- Latency requirements
- Performance capabilities of the stream processing framework

Microbatch stream processing

Spark Streaming

Process each batch in parallel



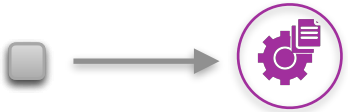
Up to 2 minutes

Example:

Sales metrics

True stream processing

Apache Flink



- Each node continuously listens to messages and updates its dependent nodes
- Delivers processed events at very low latency
- Comes with significant overhead

Example:

Security metrics

DeepLearning.AI

Data Transformations

Week 3 Summary

Batch Transformations

Considerations:

- Size of the data you need to transform
- Memory specification of the machine processing the code

Distributed processing frameworks

- Process large data that doesn't fit into memory
- Leverages parallel computation

MapReduce

Batch Transformations

Considerations:

- Size of the data you need to transform
- Memory specification of the machine processing the code

Distributed processing frameworks

- Process large data that doesn't fit into memory
- Leverages parallel computation

MapReduce

- Spark DataFrames
- Spark SQL

complex transformations

simple transformations

Batch Transformations

Considerations:

- Size of the data you need to transform
- Memory specification of the machine processing the code

Distributed processing frameworks

- Process large data that doesn't fit into memory
- Leverages parallel computation

MapReduce

complex transformations

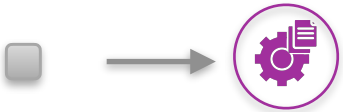
- Spark DataFrames
- Spark SQL

simple transformations

Streaming Transformations

Understand your use case and performance requirements

Apache Flink



Batch Transformations

Considerations:

- Size of the data you need to transform
- Memory specification of the machine processing the code

Distributed processing frameworks

- Process large data that doesn't fit into memory
- Leverages parallel computation

MapReduce

complex transformations

- Spark DataFrames
- Spark SQL

simple transformations

Streaming Transformations

Understand your use case and performance requirements

Apache Flink

Lab

Implement a CDC pipeline

Apache Flink