# Apache Spark
# Data Interfaces

# Resilient Distributed Dataset

- This is low-level API for working with data in Spark.
- RDDs can provide data lineage across processes as they're completed.
- RDD is a container that allows us to work with data objects.
- These objects can be of varying types and spread across many machines in the cluster.

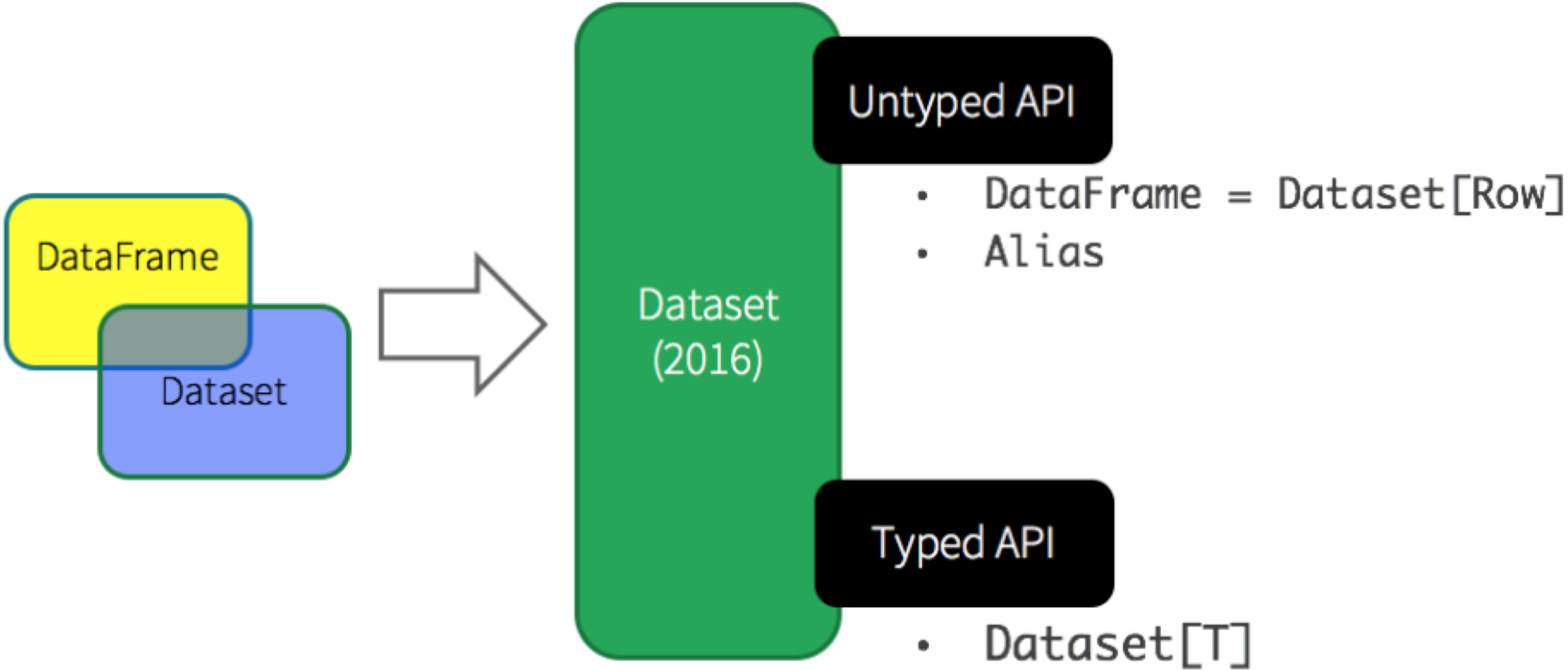
# DataFrames

- DataFrame is analogous to Pandas in Python, DataFrames in R or Tables in relational databases.
- DataFrames are based on RDD

# Dataset

- Dataset is a combination of RDDs and DataFrames.
- You can type your data like an RDD and query it like a DataFrame.

# Unified Apache Spark 2.0 API



- `DataFrame = Dataset[Row]`
- `Alias`

- `Dataset[T]`

# Directed Acyclic Graph (DAG)

- Spark features an advanced Directed Acyclic Graph (DAG) engine supporting cyclic data flow.
- Each Spark job creates a DAG of task stages to be performed on the cluster. Compared to MapReduce, which creates a DAG with two predefined stages - Map and Reduce, DAGs created by Spark can contain any number of stages.
- This allows some jobs to complete faster than they would in MapReduce, with simple jobs completing after just one stage, and more complex tasks completing in a single run of many stages, rather than having to be split into multiple jobs.

# Directed Acyclic Graph (DAG)

- Spark jobs perform work on Resilient Distributed Datasets (RDDs), an abstraction for a collection of elements that can be operated on in parallel.
- When running Spark in a Hadoop cluster, RDDs are created from files in the distributed file system in any format supported by Hadoop, such as text files, SequenceFiles, or anything else supported by a Hadoop InputFormat.
- Once data is read into an RDD object in Spark, a variety of operations can be performed by calling abstract Spark APIs.

# Directed Acyclic Graph (DAG)

- The two major types of operation available are:
  - Transformations: Transformations return a new, modified RDD based on the original. Several transformations are available through the Spark API, including map(), filter(), sample(), and union().
  - Actions: Actions return a value based on some computation being performed on an RDD. Some examples of actions supported by the Spark API include reduce(), count(), first(), and foreach().
- Some Spark jobs will require that several actions or transformations be performed on a particular data set, making it highly desirable to hold RDDs in memory for rapid access.
- Spark exposes a simple API to do this - cache(). Once this API is called on an RDD, future operations called on the RDD will return in a fraction of the time they would if retrieved from disk.

# Directed Acyclic Graph (DAG)

DAG in Apache Spark is a set of Vertices and Edges, where vertices represent the RDDs and the edges represent the Operation to be applied on RDD.

In Spark DAG, every edge is directed from earlier to later in the sequence.

On calling of Action, the created DAG is submitted to DAG Scheduler which further splits the graph into the stages of the task.

# Jump into Databricks