



Curso

Introduccitorio de Spark

Óscar Gutiérrez



Óscar Gutiérrez

- Ingeniero en Computación.
- Consultor enfocado en tecnologías de Big Data y dataviz desde 2018.
- Apasionado de NLP y desarrollo de sistemas operativos.

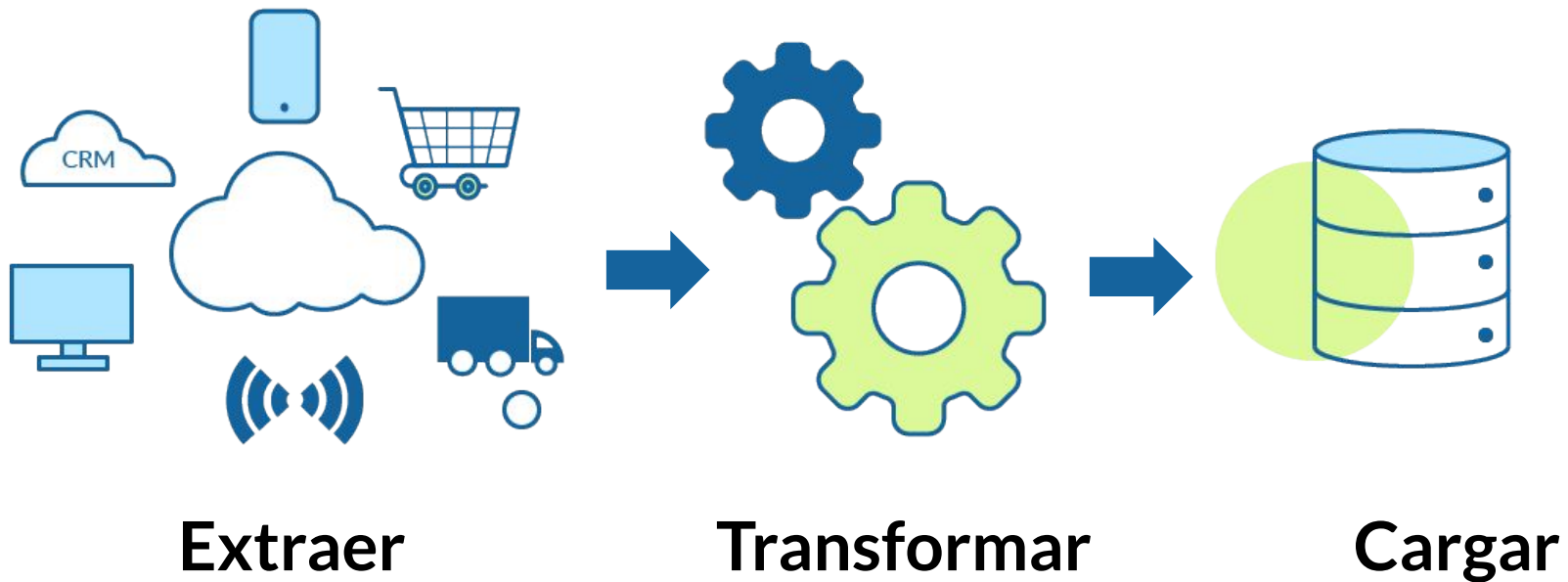
Curso de
**Introducción a
Apache Spark**

¿Qué aprenderás?



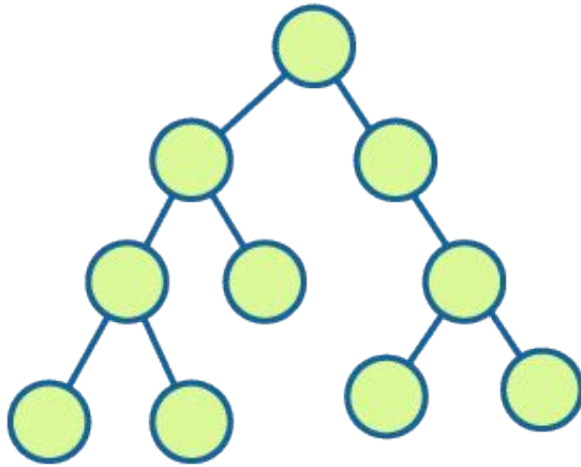
¿Qué es Apache Spark?

¿Qué aprenderás?

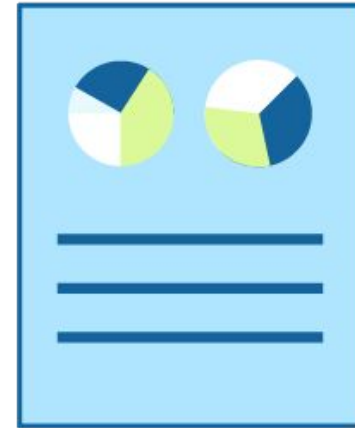


Entender un flujo de vida de ETL en Spark

¿Qué aprenderás?



**Manejo de las
estructuras base
de Spark**



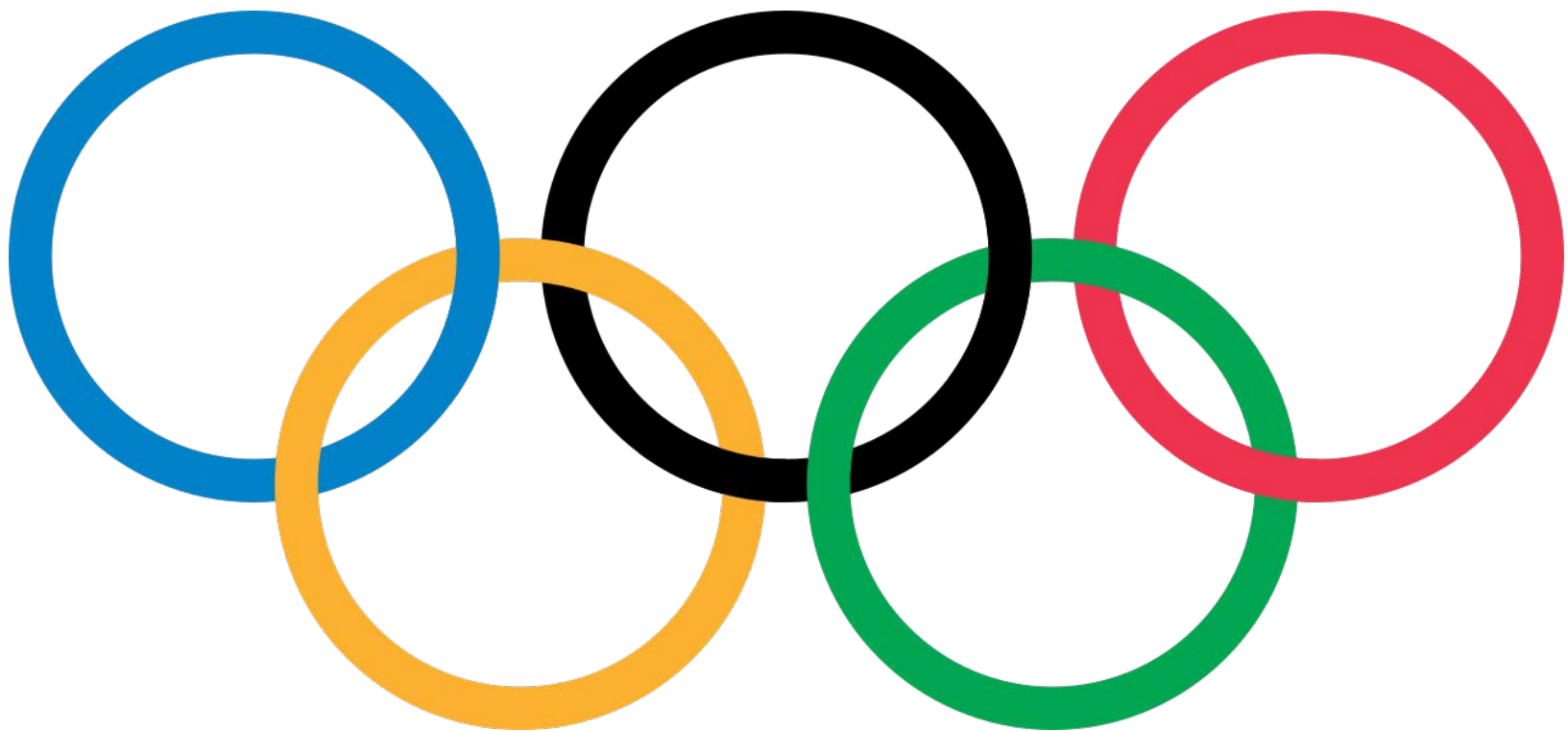
**Fundamentos de
calidad de datos**



Requisitos

- Programación orientada a objetos
- Cursos de SQL y MySQL

Manos a la obra



Curso de
**Introducción a
Apache Spark**

Introducción a Apache Spark



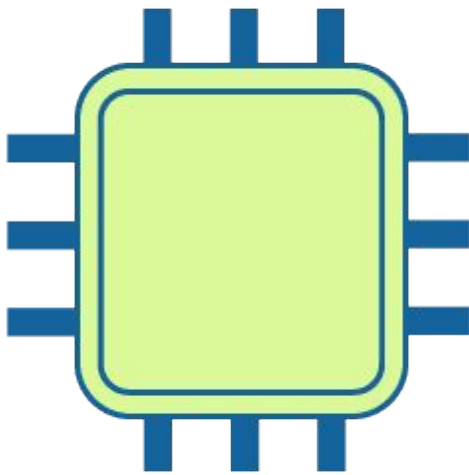
¿Qué es Spark?

- Framework de desarrollo de procesos de Big Data.
- Framework preocupado por la velocidad del proceso.

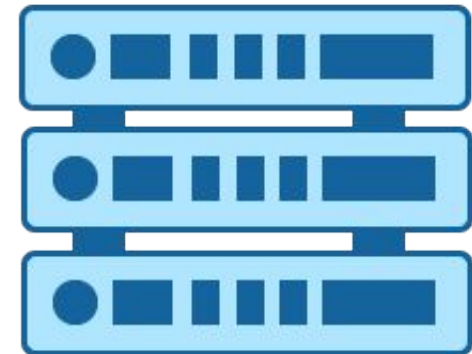
Lenguajes



¿Qué no es Apache Spark?



OLAP



OLTP



Historia

- **Nace en 2009**
Fue creado en la Universidad de Berkeley.
- **Heredera de Hadoop**
Es el paso siguiente a la tecnología Hadoop.
- **Versión 3**
Fue liberada el 10 de Junio de 2020.



Spark vs Hadoop

- Spark se enfoca en procesamiento de datos desde RAM.
- Posee naturalmente un módulo para ML, streaming y grafos.
- No depende de un sistema de archivos.

Introducción a Apache Spark

Introducción a los RDD y DataFrames



Componentes de Spark

Las dos principales estructuras que soporta Spark son los **RDD** y los **DataFrames**.

La diferencia reside en la estructura que poseen.

Los RDD son el componente mínimo con el cual podemos comunicarnos con Spark.
resilient distributed dataset



Características de los RDD

→ Principal abstracción de datos

Es la unidad básica, existen desde su inicio hasta su versión 3.0.

→ Distribución

Los RDD se distribuyen y particionan a lo largo del clúster.



Características de los RDD

→ Creación simple

Al no poseer estructura formalmente, adoptan la más intuitiva.

→ Inmutabilidad

Posterior a su creación no se pueden modificar.



Características de los RDD

→ **Ejecución perezosa**

A menos que se realice una acción.

Transformaciones y acciones

Transformaciones	Acciones
orderBy()	show()
groupBy()	take()
filter()	count()
select()	collect()
join()	save()



```
pedroParamo = sc.textFile("pedroParamo.txt")
```

```
comala = pedroParamo.filter(lambda l: "Comala" in l)
```

```
paramo = pedroParamo.filter(lambda l: "Páramo" in l)
```

```
comalaInterParamo = comala.intersection(paramo)
```

```
cuenta = comalaInterParamo.count()
```



DataFrame

→ Formato

A diferencia de un RDD poseen columnas, lo cual les otorga tipos de datos.

→ Optimización

Poseen una mejor implementación, lo cual los hace preferibles.



DataFrame

→ Facilidad de creación

Se pueden crear desde una base de datos externa, archivo o RDD existente.



¿Cuándo usar RDD?

- Cuando te interese controlar el flujo de Spark.
- Si eres usuario de Python, convertir a RDD un conjunto permite mejor control de los datos.
- Estás conectándote a versiones antiguas de Spark.

¿Cuándo usar DataFrames?

- Si poseemos semánticas de datos complicadas.
- Vamos a realizar tareas de alto nivel como filtros, mapeos, agregaciones, promedios o sumas.
- Si vamos a usar sentencias SQL-like.

Introducción a los RDD y DataFrames



CLI vs Jupyter



Transformaciones y acciones

Transformaciones sobre RDDs




Acciones de modificación a RDD

Acciones de conteo sobre RDD

Solución reto deportistas

Operaciones numéricas



Persistencia y particionado



Persistencia

Problemas al usar un RDD o DF varias veces:

- Spark recomputa el componente y sus dependencias cada vez que se ejecuta una acción.
- Es costoso (especialmente en problemas iterativos).



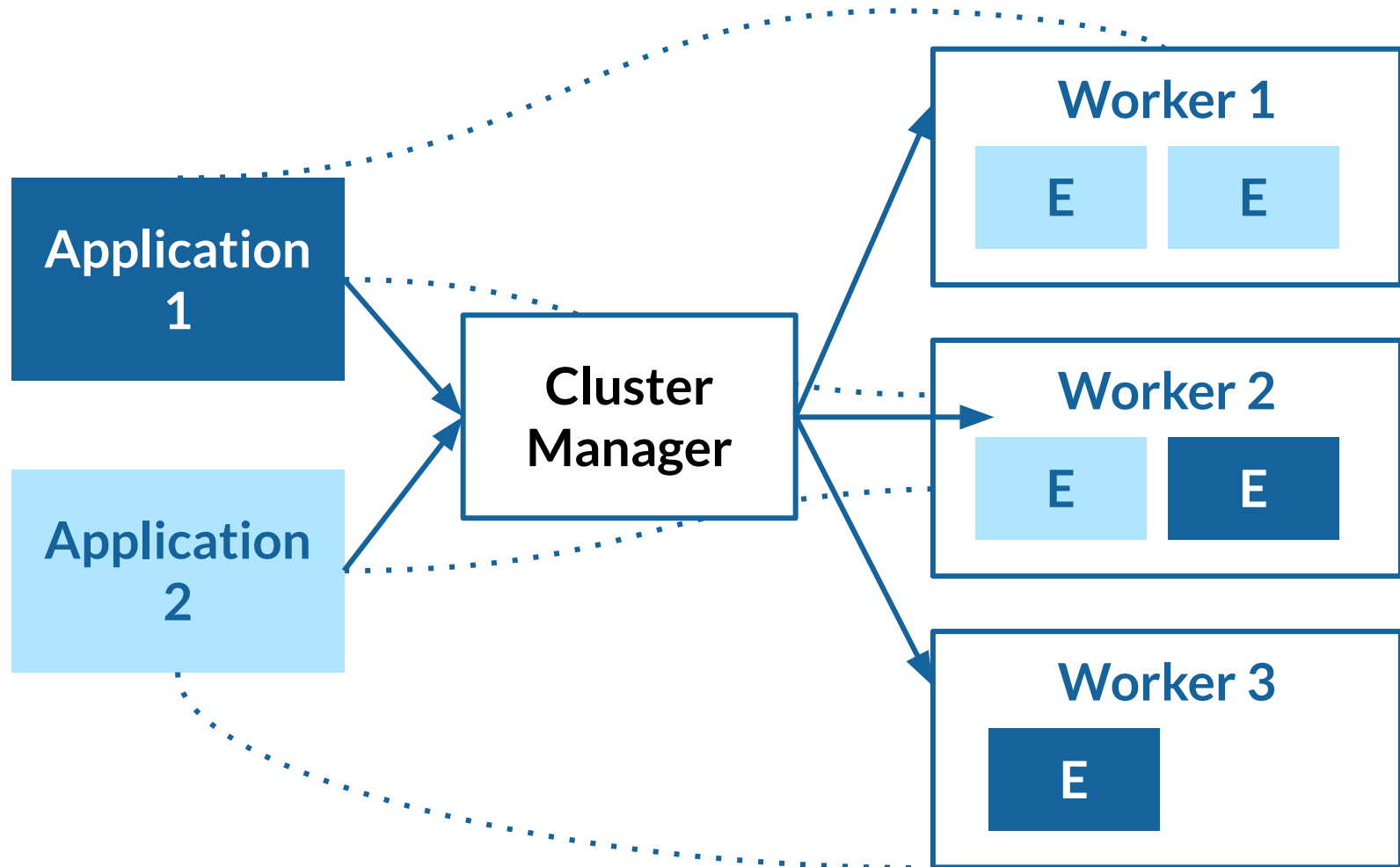
Solución

- Conservar el componente en memoria y/o disco.
- Métodos `cache()` o `persist()` nos ayudan.
- En PySpark los datos son almacenados de forma serializada.

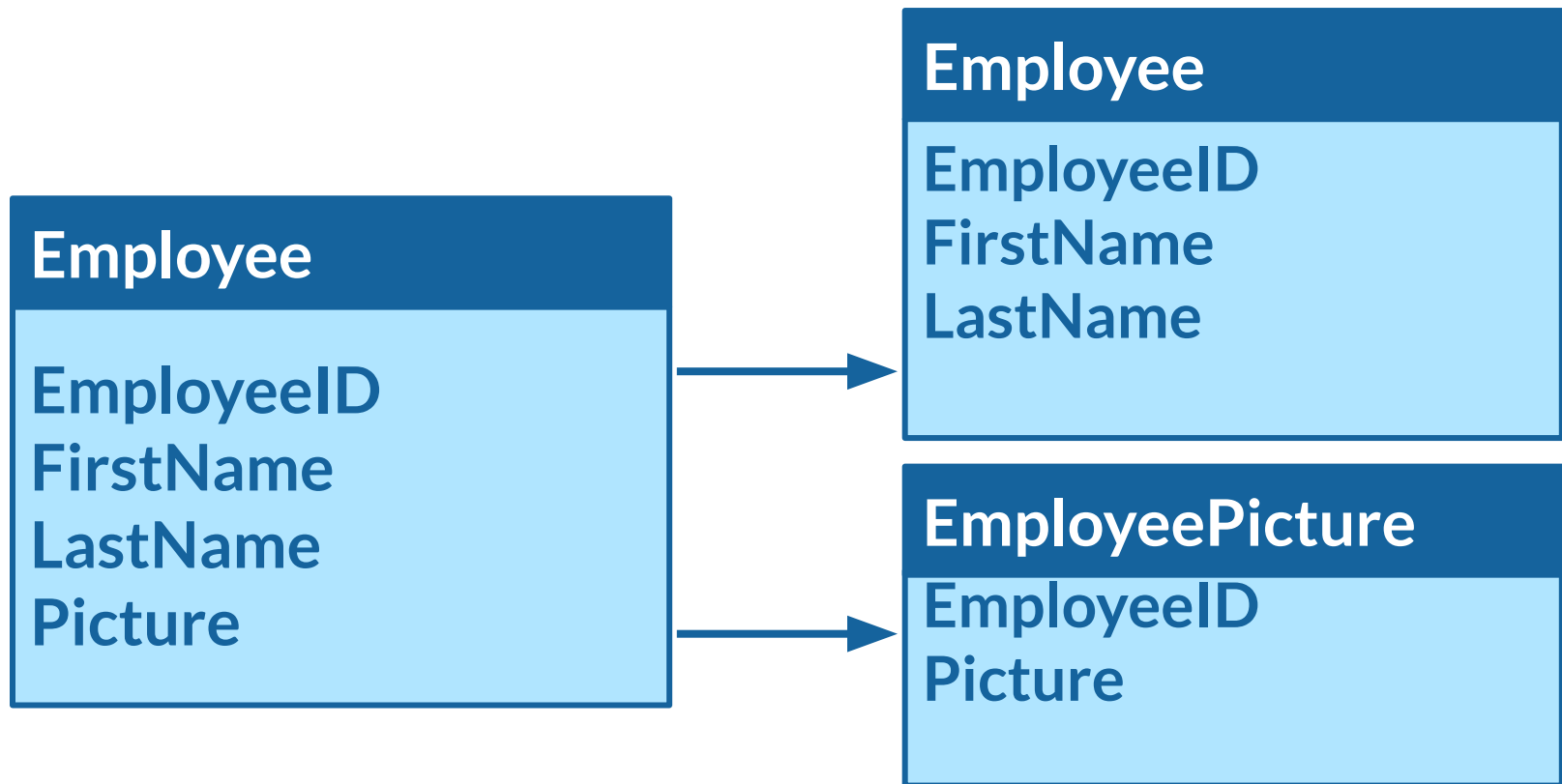
Tipos de persistencia

Nivel	Espacio	CPU	Memoria/Disco
MEMORY_ONLY	Alto	Bajo	Memoria
MEMORY_ONLY_SER	Bajo	Alto	Memoria
MEMORY_AND_SER	Alto	Medio	Memoria
MEMORY_AND_DISK	Bajo	Alto	Ambos
MEMORY_AND_DISK_SER	Bajo	Alto	Ambos
DISK_ONLY	Bajo	Alto	Disco
OFF_HEAP	Bajo	Alto	Memoria

Beneficios adicionales



Particionado



Persistencia y particionado



Particionando datos

Funciones de particionado

Lectura y escritura de archivos

Creación de DF e inferencia de tipos de datos



¿Cuáles son los beneficios?

- Permite procesar como una tabla de base de datos los DF.
- Poseen estructura y pueden ser creados como los DF.
- Una optimización superior debido al optimizador de consultas Catalyst y el motor de ejecución Tungsten.

Creación de DF e inferencia de tipos de datos



Operaciones sobre DF

Solución reto joins



Funciones de agrupación



UDF



SQL



Conclusión