

---


# Patrón de Diseño SHOW

---

# Definición de una cadena de procesos

## CÓDIGO

```
#PASO 1: TRANSFORMATION  
df1 = .....  
  
#PASO 2: TRANSFORMATION  
df2 = df1.....  
  
#PASO 3: TRANSFORMATION  
df3 = df1.....  
  
#PASO 4: TRANSFORMATION  
df4 = df1.....  
  
#PASO 5: TRANSFORMATION  
df5 = df1.....
```

- 
- Ejecutamos 5 pasos implementados con “transformations”
  - Como no se ha llamado a ningún “action”, aún no se han creado los dataframes en la RAM

## RAM



# Ejecución del action “show”

## CÓDIGO

```
#PASO 1: TRANSFORMATION
df1 = .....


#PASO 2: TRANSFORMATION
df2 = df1.....

#PASO 3: TRANSFORMATION
df3 = df1.....

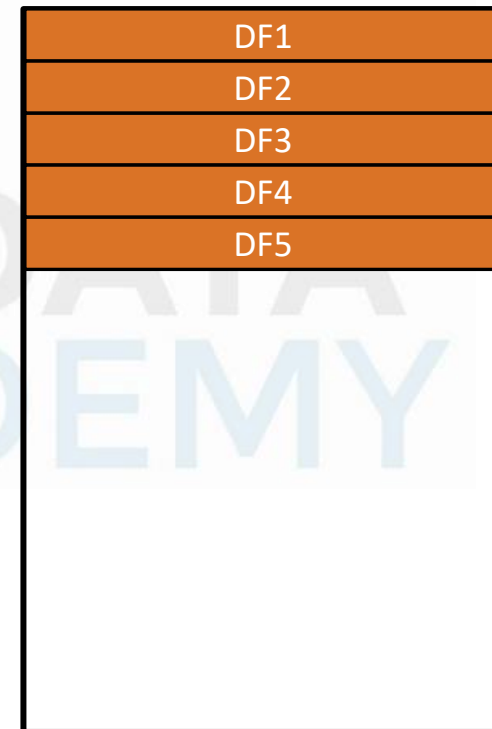
#PASO 4: TRANSFORMATION
df4 = df1.....

#PASO 5: TRANSFORMATION
df5 = df1.....

#PASO 6: ACTION
df5.show()
```

- 
- En el paso 6, ejecutamos el action “show”
  - Al ser un action, ejecuta toda la cadena de procesos que crea al “df5”
  - Supongamos que cada paso demora 1 hora en ejecutarse, en total el proceso hasta llegar al “show” toma 5 horas

## RAM



# Definición de algunos pasos más

## CÓDIGO

```
#PASO 6: ACTION
df5.show()

#PASO 7: TRANSFORMATION
df6 = df1.....

#PASO 8: TRANSFORMATION
df7 = df1.....

#PASO 9: TRANSFORMATION
df8 = df1.....

#PASO 10: TRANSFORMATION
df9 = df1.....
```



- Luego del paso 6, definimos 4 pasos de procesamiento más
- **Todos son transformations, así que en la memoria RAM aún no se crea nada**

## RAM

DF1
DF2
DF3
DF4
DF5

# El Garbage Collector llega a nuestra zona de memoria RAM

## CÓDIGO

```
#PASO 6: ACTION
df5.show()

#PASO 7: TRANSFORMATION
df6 = df1.....

#PASO 8: TRANSFORMATION
df7 = df1.....

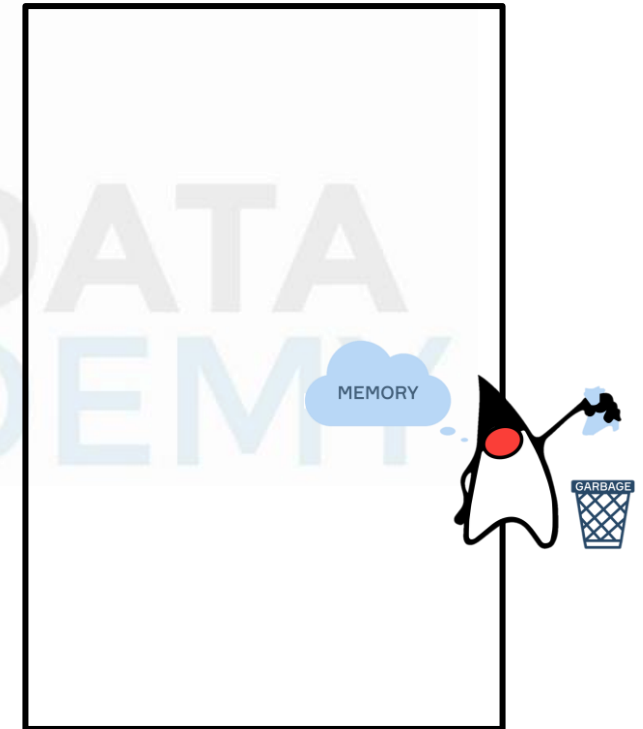
#PASO 9: TRANSFORMATION
df8 = df1.....

#PASO 10: TRANSFORMATION
df9 = df1.....
```



- 1 minuto después el GC llega a nuestra zona de memoria RAM y elimina todos los dataframes creados
- **La RAM queda vacía**

## RAM



# Ejecución del action "save"

## CÓDIGO


```
#PASO 7: TRANSFORMATION
df6 = df1.....

#PASO 8: TRANSFORMATION
df7 = df1.....

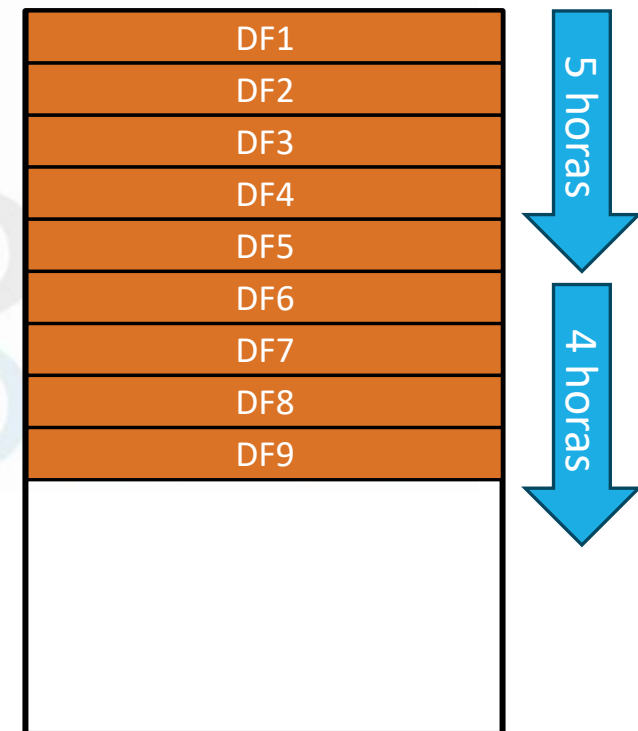
#PASO 9: TRANSFORMATION
df8 = df1.....

#PASO 10: TRANSFORMATION
df9 = df1.....

#PASO 11: ACTION
df9.write.format("csv")
```

- 
- En el paso 11, ejecutamos el action "save" para almacenar el "df9"
  - **Tendremos que esperar nuevamente 5 horas** para que se creen nuevamente los dataframes del "df1" al "df5"
  - Luego debemos esperar las 4 horas para que se creen los dataframes del "df6" al "df9"

## RAM



# ¿Cuánto tiempo tomo el proceso?

**5 HORAS** [ACTION SHOW]

DF1
DF2
DF3
DF4
DF5

RAM [EL GC BORRÓ TODO]



**9 HORAS** [ACTION SAVE]

DF1
DF2
DF3
DF4
DF5
DF6
DF7
DF8
DF9

**En total el proceso tomo 14 horas, ya que tuvimos “mala suerte”** y el GC borró algunos dataframes luego del action “show” cuando aún los necesitábamos.

# ¿Y si hubiésemos tenido “buena suerte”?

**5 HORAS** [ACTION SHOW]

DF1
DF2
DF3
DF4
DF5

**RAM** [EL GC AÚN NO LLEGA]

DF1
DF2
DF3
DF4
DF5

[El GC está lejos de nuestra zona de memoria, aún no borra nada]



**4 HORAS** [ACTION SAVE]

DF1
DF2
DF3
DF4
DF5
DF6
DF7
DF8
DF9

**En total el proceso tomo 9 horas, ya que tuvimos “buena suerte”** y el GC no borró nada



# Solución: ¿No usar los show?

```
#PASO 1: TRANSFORMATION
```

```
df1 = .....
```

```
.....
```

```
#PASO 5: TRANSFORMATION
```

```
df5 = df1.....
```

```
#PASO 6: TRANSFORMATION
```

```
df5.show()
```

```
.....
```

```
#PASO 10: TRANSFORMATION
```

```
df9 = df1.....
```

```
#PASO 11: ACTION
```

```
df9.write.format("csv")
```

- Si en el código no usamos los “show”, podemos definir una cadena de procesos con varios “transformations”, y toda la cadena se ejecutará hasta el final, cuando llamemos al “save” y se guarde en disco duro
- Si hacemos eso, el proceso siempre demorará 9 horas, ya que el GC no puede borrar dataframes de una cadena que está en ejecución
- Sin embargo, necesitamos los “show” para construir el proceso mientras lo implementamos y ver cómo va quedando cada paso del proceso

No podemos dejar de usar los show, ¿cómo solucionamos el problema?

# Solución: Patrón de Diseño SHOW

Crearemos una función que se comporte de la siguiente manera

Cuando estamos desarrollando en el Notebook

```
#Parámetro que activa y desactiva el "show"  
PARAM_SHOW_ACTIVO = True
```

```
#FUNCIÓN  
def show(df):  
    if(PARAM_SHOW_ACTIVO == True):  
        #Ejecutamos el action "show"  
        df.show()  
  
        #Lo marcamos en la caché  
        df.cache()
```

- **Con una variable, activaremos los "show"** (PARAM\_SHOW\_ACTIVO = TRUE)
- Luego de mostrarlo, lo marcamos en la "caché" de la RAM para que el GC no lo borre y el proceso no tenga que recalcular todo nuevamente

Cuando el proceso se está ejecutando en producción

```
#Parámetro que activa y desactiva el "show"  
PARAM_SHOW_ACTIVO = False
```

```
#FUNCIÓN  
def show(df):  
    if(PARAM_SHOW_ACTIVO == True):  
        #Ejecutamos el action "show"  
        df.show()  
  
        #Lo marcamos en la caché  
        df.cache()
```

- Ya no necesitamos los "show" **Con una variable los desactivamos** (PARAM\_SHOW\_ACTIVO = FALSE)