**Documentación Proyecto Spark**

Este proyecto implementa una aplicación Spark utilizando el lenguaje de programación Scala y Maven como herramienta de gestión de proyectos. La aplicación se centra en la creación y procesamiento de DataFrames mediante Spark SQL, con un enfoque especial en operaciones de join.

**Detalles del Código**

**1. Creación del Objeto SparkSession**

* Se utiliza el objeto **SparkSession** para interactuar con el entorno de Spark. En este caso, se configura con un master local de un núcleo ("local[1]").

**2. Generación de DataFrames**

* Se crean tres DataFrames (**df1**, **df2**, **df3**) con cantidades específicas de filas y columnas.

**3. Operaciones de Join**

* Se realizan operaciones de join para combinar los DataFrames de acuerdo con las especificaciones.

**4. Resultado Final**

* Se muestra en la consola el DataFrame resultante **df5**.

**Archivo llamado DF5Optimizado**

**A continuación, se explica con detalle las mejoras implementadas en el código de la app a fin de mejorar y optimizar los tiempos de carga.**

* **Configuración de Particiones**

El número de particiones en Spark afecta el rendimiento de las operaciones de shuffle, como las operaciones de join. El shuffle es la redistribución de datos entre las particiones durante ciertas operaciones, y el rendimiento puede mejorar o empeorar según cómo se configuren las particiones.

Aquí hay algunas consideraciones y beneficios:

1. **Demora de Shuffle:**
   * Un mayor número de particiones puede reducir el tamaño de los bloques de datos a transferir durante el shuffle, lo que puede acelerar la operación.
2. **Recursos del Cluster:**
   * Un mayor número de particiones puede hacer un uso más eficiente de los recursos del clúster si se tiene un clúster grande, ya que permite una mejor distribución de la carga de trabajo.

**Archivo llamado DF5Optimizado2**

* **Descripción del Cambio**

Acá detallaremos la mejora específica realizada en el código de la aplicación para optimizar el rendimiento de las operaciones de join, mediante la implementación de la técnica de "broadcasting". La optimización se centra en la generación del DataFrame df5.

* **Broadcasting en los DataFrame df4 y df5**

El broadcasting es una técnica eficiente para mejorar el rendimiento de las operaciones de join cuando un DataFrame es significativamente más pequeño que el otro. En este caso, df2 tiene 1000 filas, mientras que df1 tiene 4000 filas. Al aplicar broadcasting en df2, se minimiza el movimiento de datos durante la operación de join, lo que resulta en una ejecución más eficiente.

* **Impacto en el Rendimiento**

La aplicación de broadcasting en las operaciones de join tiene un impacto positivo significativo en el rendimiento, especialmente cuando se trabaja con DataFrames de tamaños desiguales. Reducir el movimiento de datos optimiza el tiempo de ejecución y mejora la eficiencia general del procesamiento.

**Archivo llamado DF5Optimizado3**

* **Descripción del Cambio**

Acá se presentan las mejoras específicas realizadas en el código de la aplicación para optimizar el rendimiento de las operaciones de join y particionamiento, centradas en la generación del DataFrame **df5**.

* **Particionamiento en los DataFrames de Entrada**

Se ha introducido una fase de particionamiento en los DataFrames de entrada (**df1**, **df2**, y **df3**). Esto se logra mediante la función **repartition** en cada DataFrame, ajustando el número de particiones según sea necesario.

El particionamiento se ha introducido para optimizar el rendimiento y la eficiencia de las operaciones de join. Al particionar los DataFrames, se distribuyen los datos de manera más equitativa entre los nodos del clúster, facilitando así las operaciones de join y reduciendo la necesidad de realizar shuffles extensos. Esta optimización minimiza el movimiento de datos y distribuye eficientemente la carga de trabajo entre los nodos del clúster, mejorando la velocidad de ejecución.