Lectura: Métodos de ingestión de datos por lotes y manejo de datos de diversas fuentes

Tiempo estimado necesario: 5 minutos

En esta lectura, descubrirás varios métodos de ingestión de datos por lotes en Apache Spark. También aprenderás a manejar datos de diversas fuentes.

Comencemos con los métodos de ingestión de datos por lotes.

1. Ingesta basada en archivos

¿Qué es?
La ingesta basada en archivos se refiere al proceso de importar datos desde varios formatos de archivo a Apache Spark.

¿Por qué es importante?
Entender la ingesta basada en archivos es crucial cuando trabajas con conjuntos de datos diversos almacenados en diferentes formatos.

Técnicas clave:
• Ingesta de CSV:

```
df_csv = spark.read.csv("file.csv", header=True)
df_csv.show()
```

• Ingesta de JSON:

```
df_json = spark.read.json("file.json")
df_json.show()
```

• Ingesta de Parquet:

```
df_parquet = spark.read.parquet("file.parquet")
df_parquet.show()
```

• Ingesta de cXML:
Nota: El soporte de XML puede requerir bibliotecas adicionales

```
df_xml = spark.read.format("com.databricks.spark.xml").option("rowTag", "record").load("file.xml")
df_xml.show()
```

2. Replicación de bases de datos y extracción, transformación y carga (ETL)

¿Por qué es importante?
Puedes aprovechar la replicación de bases de datos y los procesos ETL para facilitar la ingesta y transformación continua de datos.

Perspectivas clave:
• Replicación de bases de datos

```
# Assuming df_db is the DataFrame from the replicated database
df_db.write.mode("append").saveAsTable("database_table")
```

• Proceso ETL:

```
# Assuming df_raw is the raw data DataFrame
df_transformed = df_raw.select("col1", "col2").withColumn("new_col", expr("col1 + col2"))
df_transformed.write.mode("append").saveAsTable("transformed_table")
```

3. Importación de datos a través de interfaces de programación de aplicaciones (APIs)

¿Por qué es importante?
Integrar APIs externas sin problemas te permite recuperar e ingerir datos de manera eficiente.

Consideraciones clave:
• Integración de API HTTP:

```
import requests
response = requests.get("https://api.example.com/data")
json_data = response.json()
df_api = spark.read.json(spark.sparkContext.parallelize([json_data]))
df_api.show()
```

4. Protocolos de transferencia de datos

¿Qué son?
Los protocolos de transferencia de datos como el Protocolo de Transferencia de Archivos (FTP), el Protocolo de Transferencia de Archivos Seguro (SFTP) y el Protocolo de Transferencia de Hipertexto (HTTP) son esenciales para una transferencia de datos eficiente y segura.

Mejores Prácticas:
• Ingesta FTP:

```
spark.read.format("com.springml.spark.sftp").option("host", "ftp.example.com").load("/path/to/file.csv")
```

• Ingesta HTTP:

```
spark.read.text("http://example.com/data.txt")
```

Ahora, veamos varias formas en las que puedes manejar datos de diversas fuentes.

1. Evaluación de la fuente de datos

¿Por qué es importante?
Debes evaluar las características y la calidad de la fuente de datos, ya que son fundamentales para una integración de datos efectiva.

Actividades clave:
• Evaluación de características:

```
df_source.describe().show()
```

• Evaluación de calidad:

```
df_source.selectExpr("count(distinct *) as unique_records").show()
```

2. Mapeo y transformación de esquemas

¿Cuáles son los desafíos?
Mapear y transformar esquemas de datos puede ser un desafío para ti, pero al mismo tiempo, son esenciales para adaptar los datos al modelo objetivo.

Técnicas clave:
• Mapeo de esquemas:

```
df_mapped = df_raw.selectExpr("col1 as new_col1", "col2 as new_col2")
```

• Transformación de esquema:

```
df_transformed = df_mapped.withColumn("new_col3", expr("new_col1 + new_col2"))
```

3. Validación y limpieza de datos

¿Por qué es importante?
Debes garantizar una alta calidad de los datos durante la ingestión, ya que es crucial para los procesos posteriores.

Estrategias clave:
• Validación de datos:

```
df_validated = df_raw.filter("col1 IS NOT NULL AND col2 > 0")
```

• Limpieza de datos:

```
df_cleansed = df_raw.na.fill(0, ["col1"])
```

4. Deducción de datos

¿Por qué es crucial?
Debes prevenir registros duplicados durante la ingestión, ya que es vital mantener la integridad de los datos en el futuro.

Estrategias clave:
• Eliminar duplicados:

```
df_deduplicated = df_raw.dropDuplicates(["col1", "col2"])
```

5. Manejo de datos no estructurados

¿Qué son los datos no estructurados?
Los datos no estructurados incluyen documentos, imágenes, registros y más. Las técnicas para la ingestión y extracción de conocimientos son cruciales.

Técnicas avanzadas:
• **Procesamiento de Lenguaje Natural (NLP):**

```
# Using Spark NLP library for text data processing
df_text = spark.read.text("text_file.txt")
```

6. Gobernanza de datos y cumplimiento

¿Por qué es importante?
Debes asegurar prácticas de gobernanza de datos y cumplimiento con los requisitos regulatorios y las leyes de privacidad de datos, ya que son esenciales para un manejo responsable de los datos.

Prácticas clave:
• **Controles de cumplimiento:**

```
# Example: Ensure GDPR compliance
df_gdpr_compliant = df_raw.filter("country IN ('EU')")
```

Esta lectura sirve como una guía completa para navegar a través de los métodos de ingestión de datos por lotes en Spark. Puedes profundizar en los temas que se alineen con tus necesidades específicas y mejorar tus habilidades en ingeniería de datos.