Spark SQL, DataSources, DataFrame, dan Dataset APIs

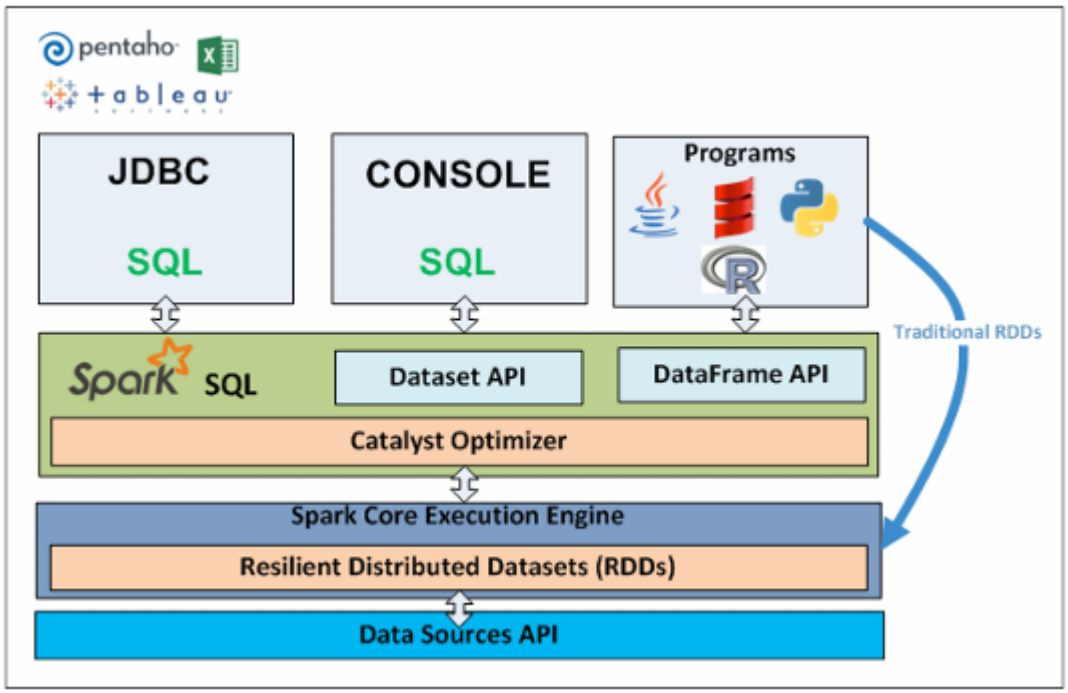
Tujuan Pembelajaran

1. Memahami konsep **Spark SQL** dan kegunaannya dalam pemrosesan data terstruktur.
2. Menguasai penggunaan **DataFrame** dan **Dataset APIs** untuk manipulasi data.
3. Mempelajari berbagai **DataSources** yang didukung Spark (JSON, CSV, Parquet, JDBC, dll.).
4. Membangun pipeline ETL sederhana menggunakan Spark SQL.

1. Pengenalan Spark SQL

Spark SQL adalah modul Apache Spark untuk pemrosesan data terstruktur. Fitur utamanya:

- **DataFrame & Dataset API**: Abstraksi data terstruktur dengan optimasi query.
- **SQL Support**: Eksekusi query SQL standar.
- **Data Source API**: Integrasi dengan berbagai format data (JSON, CSV, Parquet, Hive, JDBC, dll.).
- **Optimasi dengan Catalyst Optimizer**: Meningkatkan performa query secara otomatis.



Gambar. Arsitektur Spark SQL

2. DataFrame & Dataset APIs

Perbedaan DataFrame dan Dataset

DataFrame	Dataset
Koleksi data terdistribusi dalam bentuk *row* dengan skema	Koleksi data terdistribusi dengan tipe *strongly-typed* (Scala/Java)
Optimasi eksekusi (Tungsten, Catalyst)	Memiliki keunggulan type-safety (compile-time checking)
Dibangun di atas RDD	Hanya tersedia di Scala & Java

Contoh Pembuatan DataFrame

[1]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("DataFrameDemo").getOrCreate()

# Membuat DataFrame dari list
data = [("Alice", 25), ("Bob", 30), ("Charlie", 35)]
df = spark.createDataFrame(data, ["name", "age"])
df.show()
```

```
+-------+---+
|   name|age|
+-------+---+
|  Alice| 25|
|    Bob| 30|
|Charlie| 35|
+-------+---+
```

3. DataSources di Spark

Spark mendukung berbagai format data:

- **JSON**
- **CSV**
- **Parquet** (format kolumnar yang efisien)
- **JDBC** (koneksi database relasional)
- **Avro**, **ORC**, **Hive**, dll.

Contoh Membaca & Menulis Data

```
# Baca file CSV
df_csv = spark.read.csv("data.csv", header=True, inferSchema=True)

# Baca file JSON
df_json = spark.read.json("data.json")

# Baca dari Parquet
df_parquet = spark.read.parquet("data.parquet")

# Simpan DataFrame ke format berbeda
df.write.mode("overwrite").parquet("output.parquet")
```

4. Operasi DataFrame & SQL

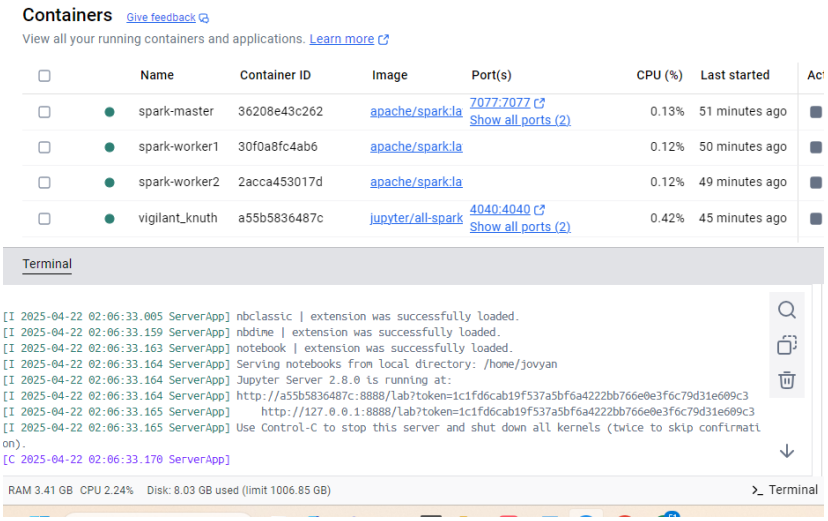
Transformasi DataFrame

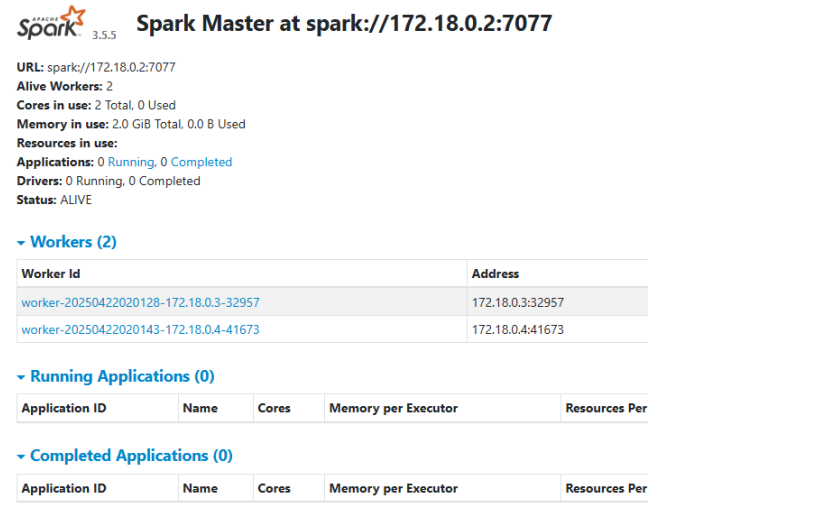
```
# Filter data
df_filtered = df.filter(df["age"] > 30)

# Select kolom
df_selected = df.select("name", "age")

# GroupBy & Aggregasi
df_grouped = df.groupBy("name").agg({"age": "avg"})

# Join DataFrame
df_join = df1.join(df2, df1["id"] == df2["id"], "inner")
```

Menggunakan SQL Query

```
# Daftarkan DataFrame sebagai temporary view

df.createOrReplaceTempView("people")



# Eksekusi query SQL

result = spark.sql("SELECT name, age FROM people WHERE age > 30")

result.show()
```

Siapkan lingkungan Spark Cluster





5. Praktikum: Membangun ETL Pipeline

Tugas

1. **Extract**: Baca data dari file CSV (sales_data.csv).
2. **Transform**:
    - Filter transaksi dengan Revenue > $100.
    - Hitung total penjualan per kategori.
3. **Load**: Simpan hasil ke Parquet.

Solusi

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, sum

spark = SparkSession.builder.appName("ETLPipeline").getOrCreate()

# Extract
df = spark.read.csv("sales_data.csv", header=True, inferSchema=True)

# Transform
df_filtered = df.filter(col("Revenue") > 100)
df_result = df_filtered.groupBy("Product_Category").agg(sum("Revenue").alias("total_sales"))

# Load
df_result.write.mode("overwrite").parquet("output_sales.parquet")

spark.stop()
```

[4]:
```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, sum

spark = SparkSession.builder.appName("ETLPipeline").getOrCreate()

# Extract
df = spark.read.csv("sales_data.csv", header=True, inferSchema=True)

# Transform
df_filtered = df.filter(col("Revenue") > 100)
df_result = df_filtered.groupBy("Product_Category").agg(sum("Revenue").alias("total_sales"))

df_result.show()

# Load
df_result.write.mode("overwrite").parquet("output_sales.parquet")

spark.stop()
```

```
+----------------+-----------+
|Product_Category|total_sales|
+----------------+-----------+
|        Clothing|    8198902|
|     Accessories|   13559164|
|           Bikes|   61782134|
+----------------+-----------+
```

6. Analisis Data Retail

Dataset

- **Format**: CSV (sales_data.csv)

Tugas

1. Hitung total pendapatan per bulan.
2. Identifikasi 5 produk terlaris.
3. Simpan hasil dalam format Parquet.

Solusi

1. Pendapatan perbulan

[8]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import month, sum, count

spark = SparkSession.builder.appName("ETLPipeline").getOrCreate()

df = spark.read.csv("sales_data.csv", header=True, inferSchema=True)

# Pendapatan per bulan
df_revenue = df.withColumn("month", month("Date")) \
               .groupBy("month") \
               .agg(sum(df["Unit_Price"] * df["Order_Quantity"]).alias("total_revenue"))

df_revenue.show()
```

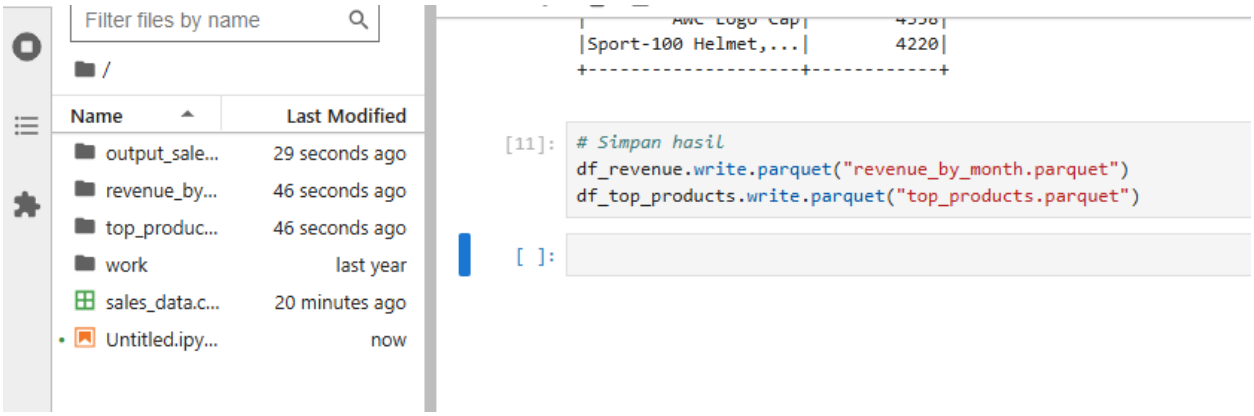
```
+-----+-------------+
|month|total_revenue|
+-----+-------------+
|   12|     10158080|
|    1|      7832338|
|    6|     10085537|
|    3|      8201790|
|    5|      9859851|
|    9|      6517880|
|    4|      8485163|
|    8|      6348349|
|    7|      6392045|
|   10|      6709394|
|   11|      6977157|
```

2. Identifikasi 5 Produk terlaris

[10]:
```
# 5 produk terlaris
df_top_products = df.groupBy("Product") \
                    .agg(count("*").alias("total_orders")) \
                    .orderBy("total_orders", ascending=False) \
                    .limit(5)

df_top_products.show()
```

```
+--------------------+------------+
|             Product|total_orders|
+--------------------+------------+
|Water Bottle - 30...|       10794|
| Patch Kit/8 Patches|       10416|
|  Mountain Tire Tube|        6816|
|        AWC Logo Cap|        4358|
|Sport-100 Helmet,...|        4220|
+--------------------+------------+
```

3. simpan dalam format parquet



7. Evaluasi

Soal Latihan

1. Baca data dari table di database MySQL anda menggunakan Spark, dengan cara berikut

```
df = spark.read.format("jdbc") \
   .option("url", "jdbc:mysql://localhost:3306/db") \
   .option("dbtable", "table_name") \
   .option("user", "user") \
   .option("password", "password") \
   .load()
```

   Baca table apa saja.

2. Buat query Spark SQL untuk menghitung Jumlah row dalam table tersebut

Kesimpulan

- Spark SQL menyediakan antarmuka terstruktur untuk pemrosesan data besar.
- DataFrame & Dataset APIs memungkinkan manipulasi data dengan sintaks mirip SQL.
- DataSources API mendukung integrasi dengan berbagai format penyimpanan.