

Spark SQL, DataSources, DataFrame, dan Dataset APIs

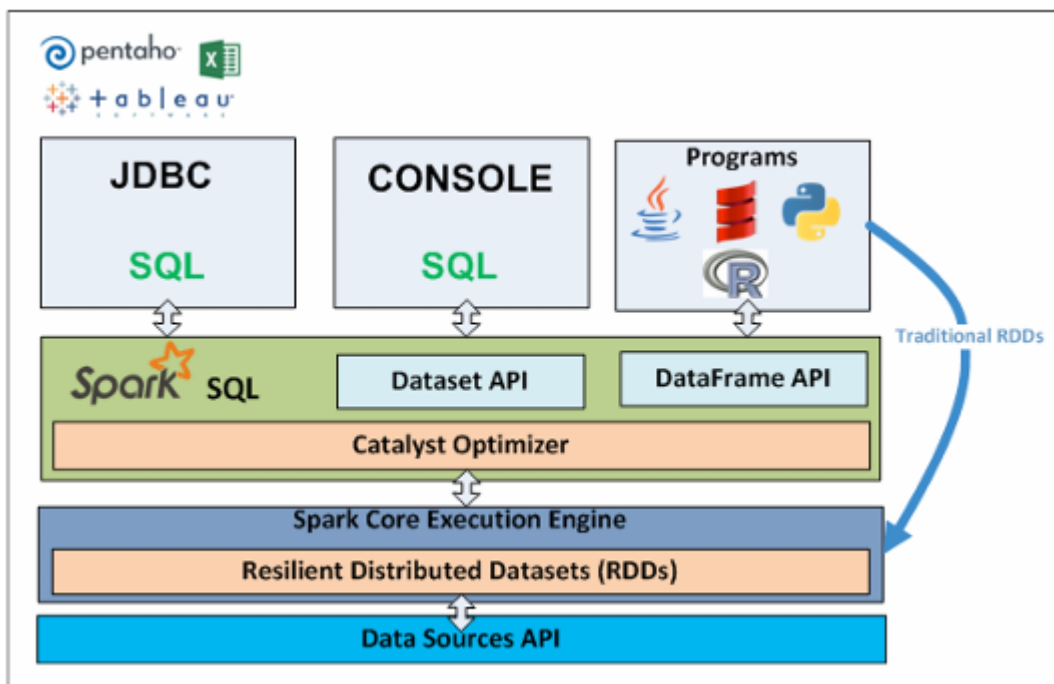
Tujuan Pembelajaran

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 <i>row</i> dengan skema	Koleksi data terdistribusi dengan tipe <i>strongly-typed</i> (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 & Agregasi
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

Containers

[Give feedback](#)

[Learn more](#)

View all your running containers and applications.

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Action
<input type="checkbox"/>	<div><div></div>spark-master</div>	36208e43c262	apache/spark:la	7077:7077 Show all ports (2)	0.13%	51 minutes ago	<div></div>
<input type="checkbox"/>	<div><div></div>spark-worker1</div>	30f0a8fc4ab6	apache/spark:la		0.12%	50 minutes ago	<div></div>
<input type="checkbox"/>	<div><div></div>spark-worker2</div>	2acca453017d	apache/spark:la		0.12%	49 minutes ago	<div></div>
<input type="checkbox"/>	<div><div></div>vigilant_knuth</div>	a55b5836487c	jupyter/all-spark	4040:4040 Show all ports (2)	0.42%	45 minutes ago	<div></div>

Terminal

[I 2025-04-22 02:06:33.005 ServerApp] nbclassic | extension was successfully loaded.
[I 2025-04-22 02:06:33.159 ServerApp] nbdlne | extension was successfully loaded.
[I 2025-04-22 02:06:33.163 ServerApp] notebook | extension was successfully loaded.
[I 2025-04-22 02:06:33.164 ServerApp] Serving notebooks from local directory: /home/jovyan
[I 2025-04-22 02:06:33.164 ServerApp] Jupyter Server 2.8.0 is running at:
[I 2025-04-22 02:06:33.164 ServerApp] http://a55b5836487c:8888/lab?token=1c1fd6cab19f537a5bf6a4222bb766e0e3f6c79d31e609c3
[I 2025-04-22 02:06:33.165 ServerApp] http://127.0.0.1:8888/lab?token=1c1fd6cab19f537a5bf6a4222bb766e0e3f6c79d31e609c3
[I 2025-04-22 02:06:33.165 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmati
on).
[C 2025-04-22 02:06:33.176 ServerApp]

RAM 3.41 GB CPU 2.24% Disk: 8.03 GB used (limit 1006.85 GB)

spark

3.5.5

Spark Master at spark://172.18.0.2:7077

URL: spark://172.18.0.2:7077

Alive Workers: 2

Cores in use: 2 Total, 0 Used

Memory in use: 2.0 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (2)

Worker Id	Address
worker-20250422020128-172.18.0.3-32957	172.18.0.3:32957
worker-20250422020143-172.18.0.4-41673	172.18.0.4:41673

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per
----------------	------	-------	---------------------	---------------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per
----------------	------	-------	---------------------	---------------

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]: 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

Filter files by name

Name	Last Modified
/	
output_sale...	29 seconds ago
revenue_by...	46 seconds ago
top_produc...	46 seconds ago
work	last year
sales_data.c...	20 minutes ago
Untitled.ipyn...	now

```
[11]: # Simpan hasil
df_revenue.write.parquet("revenue_by_month.parquet")
df_top_products.write.parquet("top_products.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.