Schema Enforcement and Schema Evaluation in PySpark

Schema enforcement and schema evaluation are essential concepts in PySpark for ensuring data integrity, consistency, and performance when working with large-scale distributed data.

---

1. Schema Enforcement (Schema on Write)

Schema enforcement refers to defining a schema explicitly while creating a DataFrame, ensuring that incoming data adheres to the expected data types and structure.

Why Schema Enforcement?

- Prevents bad or inconsistent data from entering the system.
- Optimizes performance by avoiding schema inference.
- Reduces runtime errors due to type mismatches.

Example of Schema Enforcement

```
from pyspark.sql import SparkSession

from pyspark.sql.types import StructType, StructField, IntegerType, StringType


# Initialize Spark Session

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


# Define Schema

schema = StructType([

  StructField("EmployeeID", IntegerType(), True),

  StructField("Department", StringType(), True),

  StructField("Salary", IntegerType(), True)

])


# Sample Data

data = [

  (1, "HR", 50000),

  (2, "IT", 75000),
```

```
    (3, "Finance", 62000)
]

# Create DataFrame with Schema Enforcement
df = spark.createDataFrame(data, schema=schema)

# Show DataFrame
df.printSchema()
df.show()
```

Key Points:

- The schema ensures EmployeeID is an Integer, Department is a String, and Salary is an Integer.
- If data does not conform, PySpark raises an error instead of automatically inferring data types.

---

2. Schema Evaluation (Schema on Read)

Schema evaluation occurs when data is read from external sources (CSV, JSON, Parquet, etc.), where PySpark infers or enforces the schema at runtime.

Why Schema Evaluation?

- Helps validate data before processing.
- Ensures data consistency when loading from external sources.
- Detects schema mismatches dynamically.

Example of Schema Evaluation

```
# Read a CSV file with schema inference
df_csv = spark.read.csv("employees.csv", header=True, inferSchema=True)

# Display schema
df_csv.printSchema()
```

Key Points:

- inferSchema=True allows PySpark to detect column data types automatically.

- For structured formats like Parquet and ORC, schema is preserved when writing and reading data.

Schema Mismatch Handling

If the schema of the data does not match the expected schema, Spark may:

- **Throw errors** (if strict enforcement is applied).
- **Read null values** for incompatible data types.
- **Truncate or cast values** based on compatibility.

- ## Best Practices
- ✔ Use explicit schema enforcement (`StructType`) when reading structured data for better performance.
  ✔ Use `inferSchema=True` cautiously when dealing with large datasets to avoid performance bottlenecks.
  ✔ Validate schema before processing to handle unexpected changes in data sources.