Introduction to Big Data with Spark and Hadoop

Module 4 Glossary: DataFrames and SparkSQL

Welcome! This alphabetized glossary contains many of the terms in this course. This comprehensive glossary also includes additional industry-recognized terms not used in course videos. These terms are essential for you to recognize when working in the industry, participating in user groups, and in other professional certificate programs.

Estimated reading time: 15 minutes

Term	Definition
Aggregating data	Aggregation is a Spark SQL process frequently used to present aggregated statistics. Commonly used aggregation functions such as count(), avg(), max(), and others are built into Dataframes. Users can also perform aggregation programmatically using SQL queries and table views.
Analyze data using printSchema	In this phase, users examine the schema or the DataFrame column data types using the print schema method. It is imperative to note the data types in each column. Users can apply the select() function to examine data from a specific column in detail.
Apache Spark	An in-memory and open-source application framework for distributed data processing and iterative analysis of enormous data volumes.
Catalyst phases	Catalyst analyzes the query, DataFrame, unresolved logical plan, and Catalog to create a logical plan in the Analysis phase.The logical plan evolves into an optimized logical plan in the logical optimization phase. It is the rule-based optimization step of Spark SQL. Rules such as folding, pushdown, and pruning are applicable here. Catalyst generates multiple physical plans based on the logical plan in the physical planning phase. A physical plan describes computation on datasets with specific definitions explaining how to conduct the computation. A cost model then selects the physical plan with the least cost. This explains the cost-based optimization step. Code generation is the final phase. In this phase, the Catalyst applies the selected physical plan and generates Java bytecode to run on the nodes.
Catalyst query optimization	Catalyst Optimizer uses a tree data structure and provides the data tree rule sets in the background. Catalyst performs the following four high-level tasks to optimize a query: analysis, logical optimization, physical planning, and code generation.
Catalyst	Within Spark's operational framework, it employs a pair of engines, namely Catalyst and Tungsten, in a sequential manner for query enhancement and execution. Catalyst's primary function involves deriving an optimized physical query plan from the initial logical query plan. This optimization process entails implementing a range of transformations such as predicate pushdown, column pruning, and constant folding onto the logical plan.
Cost-based optimization	Cost is measured and calculated based on the time and memory that a query consumes. Catalyst optimizer selects a query path that results in minimal time and memory consumption. As queries can use multiple paths, these calculations can become quite complex when large datasets are part of the calculation.
Creating a view in Spark SQL	It is the first step in running SQL queries in Spark SQL. It is a temporary table used to run SQL queries. Both temporary and global temporary views are supported by Spark SQL. A temporary view has a local scope. Local scope implies that the view exists within the current Spark session on the current node. A global temporary view exists within the general Spark application. This view is shareable across different Spark sessions.
DAGScheduler	As Spark acts and transforms data in the task execution processes, the DAGScheduler facilitates efficiency by orchestrating the worker nodes across the cluster. This task-tracking makes fault tolerance possible, as it reapplies the recorded operations to the data from a previous state.
DataFrame operations	Refer to a set of actions and transformations that can be applied to a DataFrame, which is a two-dimensional data structure in Spark. Data within a DataFrame is organized in a tabular format with rows and columns, similar to a table in a relational database. These operations encompass a wide range of tasks, including reading data into a DataFrame, performing data analysis, executing data transformations (such as filtering, grouping, and aggregating), loading data from external sources, and writing data to various output formats. DataFrame operations are fundamental for working with structured data efficiently in Spark.
DataFrames	Data collection is categorically organized into named columns. DataFrames are conceptually equivalent to a table in a relational database and similar to a data frame in R or Python, but with greater optimizations. They are built on top of the SparkSQL RDD API. They use RDDs to perform relational queries. Also, they are highly scalable and support many data formats and storage systems. They are developer-friendly, offering integration with most big data tooling via Spark and APIs for Python, Java, Scala, and R.
Dataset	The newest Spark data abstraction, like RDDs and DataFrames, provide APIs to access a distributed data collection. They are a collection of strongly typed Java Virtual Machine, or JVM, objects. Strongly typed implies that datasets are typesafe, and the data set's datatype is made explicit during its creation. They offer benefits of both RDDs, such as lambda functions, type-safety, and SQL Optimizations from SparkSQL.
Directed acyclic graph (DAG)	Spark uses a DAG and an associated DAGScheduler to perform RDD operations. It is a graphical structure composed of edges and vertices. Acyclic implies new edges can originate only from an existing vertex. The vertices and edges are sequential. The edges represent transformations or actions. The vertices represent RDDs. The DAGScheduler applies a graphical structure to run tasks using the RDD, performing transformation processes. DAG enables fault tolerance. Spark replicates the DAG and restores the node when a node goes down.
distinct ([numTasks]))	It helps in finding the number of varied elements in a dataset. It returns a new dataset containing distinct elements from the source dataset.
Extract, load, and transform (ELT)	It emerged because of big data processing. All the data resides in a data lake. A data lake is a pool of raw data for which the data purpose is not predefined. In a data lake, each project forms individual transformation tasks as required. It does not anticipate all the transformation requirements usage scenarios as in the case of ETL and a data warehouse. Organizations opt to use a mixture of ETL and ELT.
Extract, transform, load (ETL)	It is an important process in any data processing pipeline as the first step that provides data to warehouses for downstream applications, machine learning models, and other services.
filter (*func*)	It helps in filtering the elements of a data set basis its function. The filter operation is used to selectively retain elements from a data set or DataFrame based on a provided function (*func*). It allows you to filter and extract specific elements that meet certain criteria, making it a valuable tool for data transformation and analysis.
flatmap (*func*)	Similar to map (*func*) can map each input item to zero or more output items. Its function should return a Seq rather than a single item.
Hive tables	Spark supports reading and writing data stored in Apache Hive.

Term	Definition
Java virtual machines (JVMs)	The platform-specific component that runs a Java program. At run time, the VM interprets the Java bytecode compiled by the Java Compiler. The VM is a translator between the language and the underlying operating system and hardware.
JavaScript Object Notation (JSON)	A simplified data-interchange format based on a subset of the JavaScript programming language. IBM® Integration Bus provides support for a JSON domain. The JSON parser and serializer process messages in the JSON domain.
JSON data sets	Spark infers the schema and loads the data set as a DataFrame.
Loading or exporting the data	In the ETL pipeline's last step, data is exported to disk or loaded into another database. Also, users can write the data to the disk as a JSON file or save the data into another database, such as a Postgres (PostgresSQL) database. Users can also use an API to export data to a database, such as a Postgres database.
map (*func*)	It is an essential operation capable of expressing all transformations needed in data science. It passes each element of the source through a function *func*, thereby returning a newly formed distributed dataset.
Parquet	Columnar format that is supported by multiple data processing systems. Spark SQL allows reading and writing data from Parquet files, and Spark SQL preserves the data schema.
Python	High-level, easy-to-comprehend, interpreted, and general-purpose dynamic programming language used in code readability. It offers a robust framework that helps build quick and scalable applications for z/OS, with an ecosystem of modules to develop new applications on any platform.
RDD actions	It is used to evaluate a transformation in Spark. It returns a value to the driver program after running a computation. An example is the reduce action that aggregates the elements of an RDD and returns the result to the driver program.
RDD transformations	It helps in creating a new RDD from an existing RDD. Transformations in Spark are deemed lazy as results are not computed immediately. The results are computed after evaluation by actions. For example, map transformation passes each element of a dataset through a function. This results in a new RDD.
Read the data	When reading the data, users can load data directly into DataFrames or create a new Spark DataFrame from an existing DataFrame.
Resilient Distributed Datasets (RDDs)	A fundamental abstraction in Apache Spark that represents distributed collections of data. RDDs allow you to perform parallel and fault-tolerant data processing across a cluster of computers. RDDs can be created from existing data in storage systems (like HDFS), and they can undergo various transformations and actions to perform operations like filtering, mapping, and aggregating. The *resilient* aspect refers to the Resilient Distributed Datasets (RDDs) ability to recover from node failures, and the *distributed* aspect highlights their distribution across multiple machines in a cluster, enabling parallel processing.
R	An open-source optimized programming language for statistical analysis and data visualization. Developed in 1992, it has a robust ecosystem with complex data models and sophisticated tools for data reporting.
Scala	A programming language supporting object-oriented and functional programming. The most recent representative in the family of programming languages. Apache Spark is written mainly in Scala, which treats functions as first-class citizens. Functions in Scala can be passed as arguments to other functions, returned by other functions, and used as variables.
Schema	It is a collection of named objects. It provides a way to group those objects logically. A schema is also a name qualifier; it provides a way to use the same natural name for several objects and to prevent ambiguous references to those objects.
Spark driver program	A program that functions as software situated on the primary node of a machine. It defines operations on RDDs, specifying transformations and actions. To simplify, the Spark driver initiates a SparkContext linked to a designated Spark Master. Furthermore, it transfers RDD graphs to the Master, the location from which the standalone cluster manager operates.
Spark SQL memory optimization	The primary aim is to improve the run-time performance of a SQL query by minimizing the query time and memory consumption, thereby helping organizations save time and money.
SparkSQL	It is a Spark module that helps in structured data processing. It is used to run SQL queries on Spark DataFrames and has APIs available in Java, Scala, Python, and R.
SQL queries in SparkSQL	Spark SQL allows users to run SQL queries on Spark DataFrames.
String data type	It is the IBM® Informix® ESQL/C data type that holds character data that is null-terminated and does not contain trailing blanks.
Syntax error	If this error is detected while processing a control statement, the remaining statement is skipped and not processed. Any operands in the portion of the statement preceding the error are processed.
toDS() function	Converts data into a typed Dataset for efficient and type-safe operations in PySpark.
Transform the data	In this step of the ETL pipeline, users plan for required dataset transformations, if any. The transformation aims at retaining only the relevant data. Transformation techniques include data filtering, merging with other data sources, or performing columnar operations. Columnar operations include actions such as multiplying each column by a specific number or converting data from one unit to another. Transformation techniques can also be used to group or aggregate data. Many transformations are domain-specific data augmentation processes. The effort needed varies with the domain and the data.
Tungsten	Catalyst and Tungsten are integral components of Spark's optimization and execution framework. Tungsten is geared towards enhancing both CPU and memory performance within Spark. Unlike Java, which was initially designed for transactional applications, it seeks to bolster these aspects by employing methods more tailored to data processing within the Java Virtual Machine (JVM). To achieve optimal CPU performance, it also adopts explicit memory management, employs cache-friendly data structures through STRIDE-based memory access, supports on-demand JVM bytecode, minimizes virtual function dispatches, and capitalizes on CPU register placement and loop unrolling.

Author(s)

- Niha Ayaz Sultan

Skills Network