What is the difference between Union and UnionAll in PySpark?

Authored by

stats writer

June 24, 2024

Union and UnionAll are two methods used in PySpark to combine two or more dataframes or tables into a single dataframe. The main difference between Union and UnionAll lies in their handling of duplicate rows.

Union combines two dataframes by appending the rows of one dataframe to the other, while removing any duplicate rows that exist in both dataframes. This results in a dataframe with unique rows.

On the other hand, UnionAll simply combines the rows of both dataframes without removing any duplicates. This means that the resulting dataframe may contain duplicate rows.

In summary, Union is a function that performs a "set-like" operation, while UnionAll performs a "bag-like" operation. Union is useful when dealing with dataframes that have unique rows, while UnionAll is useful when dealing with dataframes that may have duplicate rows.

PySpark union() and unionAll() transformations are used to merge two or more DataFrame's of the same schema or structure. In this PySpark article, I will explain both union transformations with PySpark examples.

Dataframe union() - `union()` method of the DataFrame is used to merge two DataFrame's of the same structure/schema. The output includes all rows from both DataFrames and duplicates are retained. If schemas are not the same it returns an error. To deal with the DataFrames of different schemas we need to use unionByName() transformation.

Syntax

```
dataFrame1.union(dataFrame2)
```

DataFrame unionAll() - `unionAll()` is deprecated since Spark "2.0.0" version and replaced with union().

Syntax

```
dataFrame1.unionAll(dataFrame2)
```

Note: In other SQL languages, Union eliminates the duplicates but UnionAll merges two datasets including duplicate records. But, in PySpark both behave the same and recommend using DataFrame duplicate() function to remove duplicate rows.

First, let's create two DataFrame with the same schema.

First DataFrame

```
# Imports
import pyspark
from pyspark.sql import SparkSession
```

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

simpleData =

columns=
df = spark.createDataFrame(data = simpleData, schema = columns)
df.printSchema()
df.show(truncate=False)

This yields the below schema and DataFrame output.

```
# Output
root
|-- employee_name: string (nullable = true)
|-- department: string (nullable = true)
|-- state: string (nullable = true)
|-- salary: long (nullable = true)
|-- age: long (nullable = true)
|-- bonus: long (nullable = true)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James |Sales |NY |90000 |34 |10000|
|Michael |Sales |NY |86000 |56 |20000|
|Robert |Sales |CA |81000 |30 |23000|
|Maria |Finance |CA |90000 |24 |23000|
+-------------+----------+-----+------+---+-----+
```

Second DataFrame

Now, let's create a second Dataframe with the new records and some records from the above Dataframe but with the same schema.

```
# Create DataFrame2
simpleData2 =
columns2=
```

df2 = spark.createDataFrame(data = simpleData2, schema = columns2)

df2.printSchema()
df2.show(truncate=False)

This yields below output

```
# Output
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James |Sales |NY |90000 |34 |10000|
|Maria |Finance |CA |90000 |24 |23000|
|Jen |Finance |NY |79000 |53 |15000|
|Jeff |Marketing |CA |80000 |25 |18000|
|Kumar |Marketing |NY |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

Merge two or more DataFrames using union

DataFrame `union()` method merges two DataFrames and returns the new DataFrame with all rows from two Dataframes regardless of duplicate data.

```
# union() to merge two DataFrames
unionDF = df.union(df2)
unionDF.show(truncate=False)
```

As you see below it returns all records.

```
# Output
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James |Sales |NY |90000 |34 |10000|
|Michael |Sales |NY |86000 |56 |20000|
```

```
|Robert |Sales |CA |81000 |30 |23000|
|Maria |Finance |CA |90000 |24 |23000|
|James |Sales |NY |90000 |34 |10000|
|Maria |Finance |CA |90000 |24 |23000|
|Jen |Finance |NY |79000 |53 |15000|
|Jeff |Marketing |CA |80000 |25 |18000|
|Kumar |Marketing |NY |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

Merge DataFrames using unionAll

DataFrame `unionAll()` method is deprecated since PySpark "2.0.0" version and recommends using the union() method.

```
# unionAll() to merge two DataFrames
unionAllDF = df.unionAll(df2)
unionAllDF.show(truncate=False)
```

Returns the same output as above.

Merge without Duplicates

Since the union() method returns all rows without distinct records, we will use the `distinct()` function to return just one record when a duplicate exists.

```
# Remove duplicates after union() using distinct()
disDF = df.union(df2).distinct()
disDF.show(truncate=False)
```

Yields below output. As you see, this returns only distinct rows.

```
# Output
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James |Sales |NY |90000 |34 |10000|
|Maria |Finance |CA |90000 |24 |23000|
|Kumar |Marketing |NY |91000 |50 |21000|
```

```
|Michael |Sales |NY |86000 |56 |20000|
|Jen |Finance |NY |79000 |53 |15000|
|Jeff |Marketing |CA |80000 |25 |18000|
|Robert |Sales |CA |81000 |30 |23000|
+-------------+----------+-----+------+---+-----+
```

Complete Example of DataFrame Union()

```
# Imports
import pyspark
from pyspark.sql import SparkSession
```

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

simpleData =

columns=
df = spark.createDataFrame(data = simpleData, schema = columns)
df.printSchema()
df.show(truncate=False)

simpleData2 =
columns2=

df2 = spark.createDataFrame(data = simpleData2, schema = columns2)

df2.printSchema()
df2.show(truncate=False)

unionDF = df.union(df2)
unionDF.show(truncate=False)
disDF = df.union(df2).distinct()
disDF.show(truncate=False)

unionAllDF = df.unionAll(df2)
unionAllDF.show(truncate=False)

This complete example is also available at the GitHub project.

Frequently Asked Questions on union()

Can we `union()` DataFrames that have different schemas?

The `union()` can be performed on the DataFrames that have the same schema and structure. If the schemas are different we may need to use `unionByName()` or make changes to the DataFrames to align to their schemas before performing union() transformation.

Does the `union()` transformation remove duplicates?

The `union()`transformation includes all rows from both DataFrames, including duplicates. If you want to remove duplicates, you can use the `dropDuplicates` transformation after performing the `union()` or apply `distinct()` to remove the duplicates.

How does union() handle NULL values?

`union()` retains NULL values from both DataFrames. If a column has a NULL value in one DataFrame and a non-NULL value in the corresponding column of the other DataFrame, both values will be included in the result.

Can we use union() to combine DataFrames with different ordering of columns?

Yes, the `union()` transformation aligns columns based on their names, not their positions. If the columns have the same names in both DataFrames, the ordering of columns does not matter.

Conclusion

In this PySpark article, you have learned how to merge two or more DataFrame's of the same schema into a single DataFrame using the Union method and learned the unionAll() deprecates and uses duplicate() to duplicate the same elements.

Happy learning !!

Related Articles