Pridwen Datasets

Alexis Guyot

August 21, 2024

Introduction to Pridwen datasets and their functionalities through examples. Internally, Pridwen datasets remain Spark datasets/dataframes. Therefore, they benefit from the same "internal functionalities" as Spark structures: lazy evaluation, optimisation of execution plans, distributed execution, etc. Some provided functions make it easy to switch from a Spark dataset to a Pridwen dataset (and vice versa) at any time.

Additional resources:

- link to the tests source code: `https://github.com/AlexisGuyot/Pridwen/blob/master/src/test/scala/DatasetTests.scala` ;
- link to dataset impelmentation source code: `https://github.com/AlexisGuyot/Pridwen/tree/master/src/main/scala/dataset`.

Contents

1 Creating a dataset

1.1 Starting from a Spark dataset

```
val spark = SparkSession.builder.master("local").appName("PridwenDataset").getOrCreate

import spark.implicits._

case class Base(att1: String, att2: Int, att3: Boolean)
case class Multivalued(att1: String, att2: Int, att3: Boolean, att4: List[Double])
case class Nested(att4: String, att5: Int, att6: Boolean, att7: Base)

val dataset = Seq(
   Base("A", 1, true),
   Base("B", 2, false),
   Base("C", 3, true)
).toDS

val datasetM = Seq(
   Multivalued("A", 1, true, List(1.1, 1.2, 1.3)),
   Multivalued("B", 2, false, List(2.1, 2.2)),
   Multivalued("C", 3, true, List())
).toDS

val datasetN = Seq(
   Nested("A", 1, true, Base("B", 2, false)),
   Nested("C", 3, true, Base("D", 4, false)),
   Nested("E", 5, true, Base("E", 6, false))
).toDS
```

1.2 Transforming it into a Pridwen dataset

```
val data = dataset.asModel[Relation]
val dataM = datasetM.asModel[JSON]
val dataN = datasetN.asModel[JSON]

// Syntaxes alternatives
val data = Data(dataset).as[Relation]
val data = Data[Relation](dataset)
```

```
data.describe("MyRelation", true)
```

```
============= MyRelation

Model: Relation

Schema:
- att1: String
```

```
- att2: Int
- att3: Boolean

+----+----+-----+
|att1|att2| att3|
+----+----+-----+
|  A|  1| true|
|  B|  2|false|
|  C|  3| true|
+----+----+-----+

========================================
```

```
dataM.describe("MyMultiJSON", true)
```

```
============= MyMultiJSON

Model: JSON

Schema:
- att1: String
- att2: Int
- att3: Boolean
- att4: List[Double]

+----+----+-----+---------------+
|att1|att2| att3|    att4|
+----+----+-----+---------------+
|  A|  1| true|[1.1, 1.2, 1.3]|
|  B|  2|false|  [2.1, 2.2]|
|  C|  3| true|        []|
+----+----+-----+---------------+

========================================
```

```
dataN.describe("MyNestedJSON", true)
```

```
============= MyNestedJSON

Model: JSON

Schema:
- att4: String
- att5: Int
- att6: Boolean
- att7:
   - att1: String
   - att2: Int
```

```
   - att3: Boolean

+----+----+----+-------------+
|att4|att5|att6|   att7|
+----+----+----+-------------+
|  A|  1|true|{B, 2, false}|
|  C|  3|true|{D, 4, false}|
|  E|  5|true|{E, 6, false}|
+----+----+----+-------------+

=====================================
```

1.3 If needed, go back to a Spark dataset

```
   val sparkDS = data.toDS

   // Syntaxe alternative
   val sparkDS = data.toDS.withSchema[Base]
```

2 Transforming a dataset

2.1 Model change

Internally, this transformation does not modify the structure containing the dataset data. If it is a Spark dataset parameterised by a schema S (as a class), then the internal structure of the new Pridwen dataset obtained after the model change is still a Spark dataset parameterised by the class S.

```
data.as[JSON].describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att1: String
- att2: Int
- att3: Boolean

+----+----+-----+
|att1|att2| att3|
+----+----+-----+
|  A |  1 | true|
|  B |  2 |false|
|  C |  3 | true|
+----+----+-----+

=======================================
```

2.1.1 Safety brought by typing

```
dataM.as[Relation] // Does not compile because the schema of dataM does not conform to
    the relational model

data.as[Graph] // Does not compile because the schema of data does not conform to the
    graph model
```

2.2 Attribute projection

The new dataset schema obtained after projection is automatically inferred from the initial schema of the data, and the names and types of the selected attributes passed as parameters. The underlying dataset becomes a dataframe (Dataset[Row]) since Spark loses track of the schema after applying a projection. The schema of the resulting dataframe (as a StructType[1]) is updated to match the schema of the Pridwen dataset.

[1] https://spark.apache.org/docs/1.5.0/api/java/org/apache/spark/sql/types/StructType.html

2.2.1 Simple projection

```
data.select(col('att1)).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String

+----+
|att1|
+----+
|  A |
|  B |
|  C |
+----+

========================================
```

2.2.2 Projecting a nested attribute

```
dataN.select(col('att7) -> col('att2)).describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att2: Int

+----+
|att2|
+----+
|  2 |
|  4 |
|  6 |
+----+

========================================
```

2.2.3 Projecting multiple attributes

```
dataN.select(col('att4) && col('att7) -> col('att2)).describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att4: String
- att2: Int

+----+----+
|att4|att2|
+----+----+
| A |  2 |
| C |  4 |
| E |  6 |
+----+----+

=======================================
```

2.2.4 Projecting with an alias

```
data.select(col('att1).as('test)).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- test: String

+----+
|test|
+----+
|  A |
|  B |
|  C |
+----+

=======================================
```

```
dataN.select(col('att4) && (col('att7) -> col('att1)).as('test)).describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att4: String
```

```
- test: String

+----+----+
|att4|test|
+----+----+
| A |  B |
| C |  D |
| E |  E |
+----+----+

=======================================
```

2.2.5 Safety brought by typing

```
data.select(col('fail)) // Does not compile because the schema of data does not include
    an attribute named fail
```

2.3 Row selection

Internally, this transformation does not modify the structure containing the dataset data. If it is a Spark dataset parameterised by a schema S (as a class), then the internal structure of the new Pridwen dataset obtained after the selection is still a Spark dataset parameterised by the class S.

2.3.1 By comparing the values of some attributes

```
dataN.filter(col('att4) === col('att7) -> col('att1)).describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att4: String
- att5: Int
- att6: Boolean
- att7:
   - att1: String
   - att2: Int
   - att3: Boolean

+----+----+----+-------------+
|att4|att5|att6|   att7      |
+----+----+----+-------------+
|  E|  5  |true|{E, 6, false}|
+----+----+----+-------------+

=======================================
```

```
dataN.select(col('att4) && col('att7) -> col('att1))
      .filter(col('att4) === col('att1)).describe(true)
```

```
============== Dataset

Model: JSON

Schema:
- att4: String
- att1: String

+----+----+
|att4|att1|
+----+----+
| E |  E |
+----+----+

=========================================
```

2.3.2 By comparing the values of an attribute with a constant value

```
dataN.filter(col('att7) -> col('att1) === v("D")).describe(true)
```

```
============== Dataset

Model: JSON

Schema:
- att4: String
- att5: Int
- att6: Boolean
- att7:
   - att1: String
   - att2: Int
   - att3: Boolean

+----+----+----+-------------+
|att4|att5|att6|   att7      |
+----+----+----+-------------+
| C |  3 |true|{D, 4, false}|
+----+----+----+-------------+

=========================================
```

```
// Other predefined comparison operations
data.filter(col('att2) =!= v(2)).describe(true)
data.filter(col('att2) > v(2)).describe(true)
```

```
data.filter(col('att2) >= v(2)).describe(true)
data.filter(col('att2) < v(2)).describe(true)
data.filter(col('att2) <= v(2)).describe(true)
data.filter(col('att3).isNull).describe(true)
data.filter(col('att3).isNotNull).describe(true)
```

See section B.1 for the console outputs of the above transformations.

2.3.3 With multiple conditions

```
dataN.filter(
   col('att7) -> col('att1) === v("D") ||
   col('att4) === col('att7) -> col('att1)
).describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att4: String
- att5: Int
- att6: Boolean
- att7:
   - att1: String
   - att2: Int
   - att3: Boolean

+----+----+----+-------------+
|att4|att5|att6|   att7      |
+----+----+----+-------------+
| C |  3 |true|{D, 4, false}|
| E |  5 |true|{E, 6, false}|
+----+----+----+-------------+

=======================================
```

2.3.4 Comparison based on a UDF

```
// Applied on some attributes
data.filter(
   col('att1) && col('att2),
   (x: String, y: Int) => x == "A" || y == 2
).describe(true)
```

```
============= Dataset

```

```
Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean

+----+----+-----+
|att1|att2| att3|
+----+----+-----+
|  A |  1 | true|
|  B |  2 |false|
+----+----+-----+

=======================================
```

```
// Applied on all attributes
data.filter((x: String, y: Int, z: Boolean) => x == "A" || y == 2 || z).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean

+----+----+-----+
|att1|att2| att3|
+----+----+-----+
|  A |  1 | true|
|  B |  2 |false|
|  C |  3 | true|
+----+----+-----+

=======================================
```

2.3.5 Safety brought by typing

```
dataN.filter(col('att4) === col('att7) -> col('att2)) // Does not compile because
    attributes att4 and att4.att2 do not have the same type

dataN.filter(col('att7) -> col('att1) === v(1)) // Does not compile because attribute
    att7.att1 does not have the same type as value 1

data.filter(col('att1) && col('att2), (x: String, y: Int) => 0) // Does not compile
    because the return type of the UDF is not a boolean
```

```
6
7  data.filter(col('att1) && col('att2), (x: String, y: String) => true) // Does not
       compile because the type of attribute att2 (Int) is not String
8
9  data.filter(col('att1) && col('att2), (x: String, y: Int, z: Boolean) => y == 2) //
       Does not compile because the UDF has three input parameters whereas only two
       attributes are selected
10
11 dataN.filter(col('fail) === col('att7) -> col('att1)) // Does not compile because the
       attribute fail does not exist in the schema of dataN
```

2.4 Adding a new attribute

The new dataset schema obtained after adding the attribute is automatically inferred from the initial schema of the data, the name passed as a parameter and the value or return type of the operator/function used to compute the values of the new attribute. The underlying dataset becomes a dataframe (Dataset[Row]) since Spark loses track of the schema after adding a new attribute. The schema of the resulting dataframe (as a StructType[2]) is updated to match the schema of the Pridwen dataset.

2.4.1 New attribute with constant values

```
1  data.add(col('test), 0).keepModel.describe(true)
```

```
1  ============== Dataset
2
3  Model: Relation
4
5  Schema:
6  - att1: String
7  - att2: Int
8  - att3: Boolean
9  - test: Int
10
11 +----+----+-----+----+
12 |att1|att2| att3|test|
13 +----+----+-----+----+
14 |  A |  1 | true| 0  |
15 |  B |  2 |false| 0  |
16 |  C |  3 | true| 0  |
17 +----+----+-----+----+
18
19 ========================================
```

```
1  // Nested addition
2  dataN.add(col('att7) -> col('test), 0).keepModel.describe(true)
```

[2] https://spark.apache.org/docs/1.5.0/api/java/org/apache/spark/sql/types/StructType.html

```
============= Dataset

Model: JSON

Schema:
- att4: String
- att5: Int
- att6: Boolean
- att7:
   - att1: String
   - att2: Int
   - att3: Boolean
   - test: Int

+----+----+----+----------------+
|att4|att5|att6|     att7       |
+----+----+----+----------------+
|  A |  1 |true|{B, 2, false, 0}|
|  C |  3 |true|{D, 4, false, 0}|
|  E |  5 |true|{E, 6, false, 0}|
+----+----+----+----------------+

========================================
```

```
// Addition requiring a model change
data.add(col('test), List[Int](0)).changeModel[JSON].describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att1: String
- att2: Int
- att3: Boolean
- test: List[Int]

+----+----+-----+----+
|att1|att2| att3|test|
+----+----+-----+----+
|  A |  1 | true| [0]|
|  B |  2 |false| [0]|
|  C |  3 | true| [0]|
+----+----+-----+----+

========================================
```

2.4.2 New attribute with values computed from the values of other attributes

```
data.add(col('test), col('att2) + v(10)).keepModel.describe(true)

// Other operations than + that are predefined: -, *, %, /
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean
- test: Int

+----+----+-----+----+
|att1|att2| att3|test|
+----+----+-----+----+
|  A |  1 | true| 11 |
|  B |  2 |false| 12 |
|  C |  3 | true| 13 |
+----+----+-----+----+

========================================
```

```
dataN.add(col('test), col('att5) + (col('att7) -> col('att2))).keepModel.describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att4: String
- att5: Int
- att6: Boolean
- att7:
   - att1: String
   - att2: Int
   - att3: Boolean
- test: Int

+----+----+----+-------------+----+
|att4|att5|att6|   att7      |test|
+----+----+----+-------------+----+
|  A |  1 |true|{B, 2, false}| 3  |
|  C |  3 |true|{D, 4, false}| 7  |
|  E |  5 |true|{E, 6, false}| 11 |
+----+----+----+-------------+----+
```

```
==========================================
```

2.4.3 New attribute with values computed with a UDF

```
// From a single attribute (without nesting)
data.add(col('test), col('att2), (x: Int) => x + 1).keepModel.describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean
- test: Int

+----+----+-----+----+
|att1|att2| att3|test|
+----+----+-----+----+
|  A |  1 | true|  2 |
|  B |  2 |false|  3 |
|  C |  3 | true|  4 |
+----+----+-----+----+

==========================================
```

```
// From a single attribute (with nesting)
dataN.add(
   col('att7) -> col('test),
   col('att7) -> col('att2),
   (x: Int) => x + 1
).keepModel.describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att4: String
- att5: Int
- att6: Boolean
- att7:
   - att1: String
   - att2: Int
   - att3: Boolean
```

```
   - test: Int

+----+----+----+----------------+
|att4|att5|att6|     att7       |
+----+----+----+----------------+
|  A |  1 |true|{B, 2, false, 3}|
|  C |  3 |true|{D, 4, false, 5}|
|  E |  5 |true|{E, 6, false, 7}|
+----+----+----+----------------+

========================================
```

```
// From multiple attributes
data.add(
   col('test), col('att1) && col('att2),
   (x: String, y: Int) => s"$x$y"
).keepModel.describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean
- test: String

+----+----+-----+----+
|att1|att2| att3|test|
+----+----+-----+----+
|  A |  1 | true| A1 |
|  B |  2 |false| B2 |
|  C |  3 | true| C3 |
+----+----+-----+----+

========================================
```

```
// From all attributes
data.add(col('test), "*", (x: String, y: Int, z: Boolean) => 0).keepModel.describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
```

```
- att3: Boolean
- test: Int

+----+----+-----+----+
|att1|att2| att3|test|
+----+----+-----+----+
|  A |  1 | true|  0 |
|  B |  2 |false|  0 |
|  C |  3 | true|  0 |
+----+----+-----+----+

=======================================
```

2.4.4 Safety brought by typing

```
data.add(col('test), List[Int]()).keepModel // Does not compile because a multivalued
    attribute cannot be added to a relation

data.add(col('test), col('att1) + v(10)).keepModel // Does not compile because att1 is
    not a numerical attribute

data.add(col('test), col('att1) + col('att2)).keepModel // Same, only att2 is a
    numerical attribute

data.add(col('test), col('fail) + v(10)).keepModel // Does not compile because
    attribute fail does not exist in the schema of data

data.add(col('test), col('att1) && col('att2), (x: String, y: String) =>
    s"$x$y").keepModel // Does not compile because the signature of the UDF passed as
    parameter is erroneous (the type of att2 is Int whereas y is a String)
```

2.5 Deleting an attribute

The new dataset schema obtained after deleting the attribute is automatically inferred from the initial schema of the data, from which the attribute whose name is passed as a parameter is removed. The underlying dataset becomes a dataframe (Dataset[Row]) since Spark loses track of the schema after deleting an attribute. The schema of the resulting dataframe (as a StructType[3]) is updated to match the schema of the Pridwen dataset.

```
data.drop(col('att1)).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
```

[3] https://spark.apache.org/docs/1.5.0/api/java/org/apache/spark/sql/types/StructType.html

```
- att2: Int
- att3: Boolean

+----+-----+
|att2| att3|
+----+-----+
|  1 | true|
|  2 |false|
|  3 | true|
+----+-----+

=======================================
```

2.5.1 Safety brought by typing

```
data.drop(col('fail)) // Does not compile because attribute fail does not exist in the
    schema of data
```

2.6 Attribute update

The new dataset schema obtained after updating the attribute is automatically inferred from the initial schema of the data, the attribute names passed as parameters and, if applicable, the return type of the function used to modify the attribute values. The underlying dataset becomes a dataframe (Dataset[Row]) since Spark loses track of the schema after updating an attribute. The schema of the resulting dataframe (as a StructType[4]) is updated to match the schema of the Pridwen dataset.

2.6.1 Renaming an attribute

```
data.withColumnRenamed('att1, 'test).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- test: String
- att2: Int
- att3: Boolean

+----+----+-----+
|test|att2| att3|
+----+----+-----+
|  A |  1 | true|
|  B |  2 |false|
|  C |  3 | true|
```

[4] https://spark.apache.org/docs/1.5.0/api/java/org/apache/spark/sql/types/StructType.html

```
+----+----+-----+

=========================================
```

2.6.2 Modifying the values of an attribute

```
// With name change
data.update('att1, 'test, (x: String) => 0).keepModel.describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- test: Int
- att2: Int
- att3: Boolean

+----+----+-----+
|test|att2| att3|
+----+----+-----+
|  0 |  1 | true|
|  0 |  2 |false|
|  0 |  3 | true|
+----+----+-----+

=========================================
```

```
// Without name change
data.update('att1, (x: String) => 0).keepModel.describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: Int
- att2: Int
- att3: Boolean

+----+----+-----+
|att1|att2| att3|
+----+----+-----+
|  0 |  1 | true|
|  0 |  2 |false|
|  0 |  3 | true|
+----+----+-----+
```

```
17
18 ========================================
```

2.6.3 Safety brought by typing

```
1 data.update('fail, 'test, (x: String) => 0).keepModel // Does not compile because
      attribute fail does not exist in the schema of data
2
3 data.update('att1, 'test, (x: Int) => 0).keepModel // Does not compile because the
      signature of the UDF passed as parameter is erroneous (the type of att1 is String
      whereas x is an Int)
```

2.7 Sorting a dataset

Internally, this transformation does not modify the structure containing the dataset data. If it is a Spark dataset parameterised by a schema S (as a class), then the internal structure of the new Pridwen dataset obtained after the sorting is still a Spark dataset parameterised by the class S.

```
1 // Only one ordering criterion. Supported orders: asc (ascending), desc (descending)
2 data.orderBy(col('att1).desc).describe(true)
```

```
1 ============= Dataset
2
3 Model: Relation
4
5 Schema:
6 - att1: String
7 - att2: Int
8 - att3: Boolean
9
10 +----+----+-----+
11 |att1|att2| att3|
12 +----+----+-----+
13 |  C |  3 | true|
14 |  B |  2 |false|
15 |  A |  1 | true|
16 +----+----+-----+
17
18 ========================================
```

```
1 // Multiple ordering criteria
2 data.orderBy(col('att1).desc && col('att2).asc).describe(true)
```

```
1 ============= Dataset
2
3 Model: Relation
```

```
Schema:
- att1: String
- att2: Int
- att3: Boolean

+----+----+-----+
|att1|att2| att3|
+----+----+-----+
|  C |  3 | true|
|  B |  2 |false|
|  A |  1 | true|
+----+----+-----+

=======================================
```

2.7.1 Safety brought by typing

```
data.orderBy(col('fail).desc) // Does not compile because attribute fail does not exist
    in the schema of data
```

2.8 Joins between datasets

The new dataset schema obtained after the join is automatically inferred by merging the original data schemas. The underlying dataset becomes a dataframe (Dataset[Row]) since Spark loses track of the schema after a join. The schema of the resulting dataframe (as a StructType[5]) is updated to match the schema of the Pridwen dataset.

```
// New dataset creation
val data2 = data.drop(col('att3))
            .add(col('test), col('att2) % v(2))
            .keepModel.drop(col('att2))
            .withColumnRenamed('att1, 'truc)
data2.describe(true) // Relation
```

```
============= Dataset

Model: Relation

Schema:
- truc: String
- test: Int

+----+----+
|truc|test|
+----+----+
```

[5] https://spark.apache.org/docs/1.5.0/api/java/org/apache/spark/sql/types/StructType.html

```
|  A | 1 |
|  B | 0 |
|  C | 1 |
+----+----+

=======================================
```

```
// Inner join with a single condition
data.join(data2, col('att1) === col('truc)).keepLeftModel.describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean
- truc: String
- test: Int

+----+----+-----+----+----+
|att1|att2| att3|truc|test|
+----+----+-----+----+----+
|  A |  1 | true|  A |  1 |
|  B |  2 |false|  B |  0 |
|  C |  3 | true|  C |  1 |
+----+----+-----+----+----+

=======================================
```

```
// Inner join with multiple conditions preserving the left dataset model (data).
    Supports the same comparison operations as filter
data.join(
   data2,
   col('att1) === col('truc) && col('att2) > col('test)
).keepLeftModel.describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean
- truc: String
- test: Int
```

```
+----+----+-----+----+----+
|att1|att2| att3|truc|test|
+----+----+-----+----+----+
|  B |  2 |false|  B |  0 |
|  C |  3 | true|  C |  1 |
+----+----+-----+----+----+

========================================
```

```
// Inner join preserving the right dataset model (dataM)
data.join(dataM, col('att1) === col('att1)).keepRightModel.describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att1: String
- att2: Int
- att3: Boolean
- att1: String
- att2: Int
- att3: Boolean
- att4: List[Double]

+----+----+-----+----+----+-----+---------------+
|att1|att2| att3|att1|att2| att3| att4          |
+----+----+-----+----+----+-----+---------------+
|  A |  1 | true|  A |  1 | true|[1.1, 1.2, 1.3]|
|  B |  2 |false|  B |  2 |false|  [2.1, 2.2]   |
|  C |  3 | true|  C |  3 | true|        []     |
+----+----+-----+----+----+-----+---------------+

========================================
```

```
// Inner join requiring a model change
data.join(dataM, col('att1) === col('att1)).changeModel[JSON].describe(true)
```

```
============= Dataset

Model: JSON

Schema:
- att1: String
- att2: Int
- att3: Boolean
- att1: String
```

```
- att2: Int
- att3: Boolean
- att4: List[Double]

+----+----+-----+----+----+-----+---------------+
|att1|att2| att3|att1|att2| att3| att4          |
+----+----+-----+----+----+-----+---------------+
|  A |  1 | true|  A |  1 | true|[1.1, 1.2, 1.3]|
|  B |  2 |false|  B |  2 |false|  [2.1, 2.2]   |
|  C |  3 | true|  C |  3 | true|        []     |
+----+----+-----+----+----+-----+---------------+

=========================================
```

```
// With an explicit mention of the join mode. Supported modes: inner, cross, outer,
    full, fullouter, full_outer, left, leftouter, left_outer, right, rightouter,
    right_outer (all aside semi and anti joins, for now)
data.join(data2, col('att1) === col('truc), "left").keepLeftModel.describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean
- truc: String
- test: Int

+----+----+-----+----+----+
|att1|att2| att3|truc|test|
+----+----+-----+----+----+
|  A |  1 | true|  A |  1 |
|  B |  2 |false|  B |  0 |
|  C |  3 | true|  C |  1 |
+----+----+-----+----+----+

=========================================
```

2.8.1 Safety brought by typing

```
data.join(data2, col('att1) === col('truc), "fail").keepLeftModel // Does not compile
    because "fail" is not a supported join mode

data.join(dataM, col('att1) === col('att1)).keepLeftModel // Does not compile because
    the result would be a relation (keepLeftModel) including a multivalued attribute
    (att4 from dataM)
```

```
4
5 data.join(data2, col('fail) === col('truc)).keepLeftModel // Does not compile because
      attribute fail does not exist in the schema of data
```

2.9 Aggregation of a dataset

The new dataset schema obtained after the aggregation is automatically inferred from the initial schema of the data, the attribute names passed as parameters to the groupBy function, and the return types of the operators passed as parameters to the agg function. The underlying dataset becomes a dataframe (Dataset[Row]) since Spark loses track of the schema after applying an aggregation. The schema of the resulting dataframe (as a StructType[6]) is updated to match the schema of the Pridwen dataset.

```
1 data.groupBy(col('att3))
2    .agg(col('att3).count && col('att2).avg && col('att2).max)
3    .keepModel.describe(true)
```

```
1  ============= Dataset
2
3  Model: Relation
4
5  Schema:
6  - att3: Boolean
7  - max: Int
8  - avg: Double
9  - count: Long
10
11 +-----+-----+---+---+
12 | att3|count|avg|max|
13 +-----+-----+---+---+
14 | true|  2  |2.0| 3 |
15 |false|  1  |2.0| 2 |
16 +-----+-----+---+---+
17
18 ========================================
```

2.9.1 Safety brought by typing

```
1 data.groupBy(col('att3)).agg(col('att1).avg).keepModel // Does not compile because att1
      is not a numerical attribute whereas the avg operator can only be applied on this
      kind of attributes
2
3 data.groupBy(col('fail)).agg(col('fail).count).keepModel // Does not compile because
      attribute fail does not exist in the schema of data
```

[6] https://spark.apache.org/docs/1.5.0/api/java/org/apache/spark/sql/types/StructType.html

A Complete example

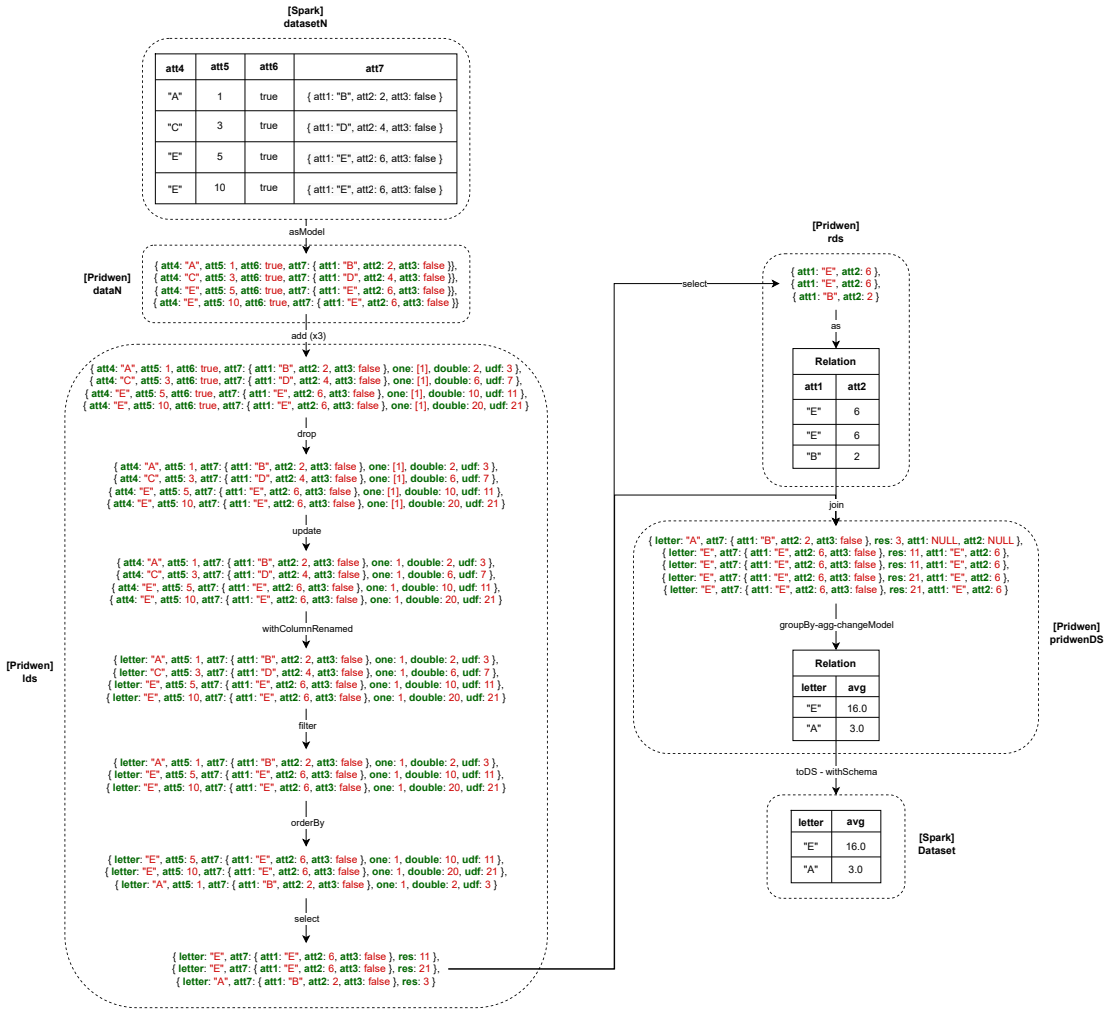

Figure 1: Applied transformations

```
val datasetN = Seq(
   SchemaDepartNested("A", 1, true, SchemaDepart("B", 2, false)),
   SchemaDepartNested("C", 3, true, SchemaDepart("D", 4, false)),
   SchemaDepartNested("E", 5, true, SchemaDepart("E", 6, false)),
   SchemaDepartNested("E", 10, true, SchemaDepart("E", 6, false))
).toDS

val dataN = datasetN.asModel[JSON]

```

```
val lds = dataN.add(col('one), List[Int](1)).keepModel
   .add(col('double), col('att5) * v(2)).keepModel
   .add(col('udf), col('double) && col('one) && col('att6), (x: Int, y: List[Int], z:
       Boolean) => if(z) x + y(0) else x - y(0)).keepModel
   .drop(col('att6))
   .update('one, (x: List[Int]) => x(0)).keepModel
   .withColumnRenamed('att4, 'letter)
   .filter(col('letter) === col('att7) -> col('att1) || col('udf) === v(3))
   .orderBy(col('letter).desc)
   .select(col('letter) && col('att7) && col('udf).as('res))
lds.describe(true)

val rds = lds.select(col('att7) -> col('att1) && col('att7) -> col('att2)).as[Relation]
rds.describe(true)

val pridwenDS = lds.join(rds, col('letter) === col('att1), "left").keepLeftModel
   .groupBy(col('letter))
   .agg(col('res).avg)
   .changeModel[Relation]
pridwenDS.describe(true)

case class FinalSchema(letter: String, avg: Double)
pridwenDS.toDS.withSchema[FinalSchema].show
```

Applied transformations (figure 1):

1. starting from a Spark dataset ;
2. transforming it into a Pridwen dataset ;
3. adding a new attribute (x3) ;
4. deleting an attribute ;
5. modifying the values of an attribute ;
6. renaming an attribute ;
7. rows selection ;
8. sorting a dataset ;
9. attribute projection (x2) ;
10. model change ;
11. join between datasets ;
12. aggregation of a dataset ;
13. go back to a Spark dataset.

```
============= Dataset

Model: JSON

Schema:
- letter: String
- att7:
   - att1: String
   - att2: Int
   - att3: Boolean
- res: Int

+------+-------------+---+
|letter|     att7    |res|
+------+-------------+---+
|   E  |{E, 6, false}| 11|
|   E  |{E, 6, false}| 21|
|   A  |{B, 2, false}| 3 |
+------+-------------+---+

=======================================

============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int

+----+----+
|att1|att2|
+----+----+
| E  |  6 |
| E  |  6 |
| B  |  2 |
+----+----+

=======================================

============= Dataset

Model: Relation

Schema:
- letter: String
- avg: Double

+------+----+
|letter| avg|
+------+----+
```

```
|    E |16.0|
|    A | 3.0|
+------+----+

========================================

+------+----+
|letter| avg|
+------+----+
|    E |16.0|
|    A | 3.0|
+------+----+
```

B Other console outputs

B.1 Row(s) selection

```
data.filter(col('att2) =!= v(2)).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean

+----+----+----+
|att1|att2|att3|
+----+----+----+
|  A |  1 |true|
|  C |  3 |true|
+----+----+----+

========================================
```

```
data.filter(col('att2) > v(2)).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean
```

```

+----+----+----+
|att1|att2|att3|
+----+----+----+
|  C |  3 |true|
+----+----+----+

========================================
```

```
data.filter(col('att2) >= v(2)).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean

+----+----+-----+
|att1|att2| att3|
+----+----+-----+
|  B |  2 |false|
|  C |  3 | true|
+----+----+-----+

========================================
```

```
data.filter(col('att2) < v(2)).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean

+----+----+----+
|att1|att2|att3|
+----+----+----+
|  A |  1 |true|
+----+----+----+

========================================
```

```
data.filter(col('att2) <= v(2)).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean

+----+----+-----+
|att1|att2| att3|
+----+----+-----+
|  A |  1 | true|
|  B |  2 |false|
+----+----+-----+

========================================
```

```
data.filter(col('att3).isNull).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
- att3: Boolean

+----+----+----+
|att1|att2|att3|
+----+----+----+
+----+----+----+

========================================
```

```
data.filter(col('att3).isNotNull).describe(true)
```

```
============= Dataset

Model: Relation

Schema:
- att1: String
- att2: Int
```

```
- att3: Boolean

+----+----+-----+
|att1|att2| att3|
+----+----+-----+
|  A |  1 | true|
|  B |  2 |false|
|  C |  3 | true|
+----+----+-----+

=======================================
```