Datasets de Pridwen

Alexis Guyot

21 août 2024

Présentation par l'exemple des datasets de Pridwen et de leurs fonctionnalités. En interne, les datasets de Pridwen restent des datasets/dataframes de Spark. Ils profitent donc des mêmes "fonctionnalités internes" que ces derniers : lazy evaluation, optimisation des plans d'exécution, exécution distribuée, etc. Des fonctions permettent de facilement passer d'un dataset de Spark à un dataset de Pridwen — et vice versa — à tout moment.

Ressources additionnelles :
— lien vers le code source des tests : `https://github.com/AlexisGuyot/Pridwen/blob/master/src/test/scala/DatasetTests.scala` ;
— lien vers le code source de l'implémentation des datasets : `https://github.com/AlexisGuyot/Pridwen/tree/master/src/main/scala/dataset`.

Table des matières

1 Créer un dataset

1.1 Partir d'un dataset de Spark

```
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 Le transformer en dataset de Pridwen

```
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 Au besoin, revenir à un dataset de Spark

```
   val sparkDS = data.toDS

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

2 Transformer un dataset

2.1 Changer de modèle

En interne, cette transformation ne modifie pas la structure contenant les données du dataset. S'il s'agit d'un dataset de Spark paramétré par un schéma S (sous forme de classe), alors la structure interne du nouveau dataset de Pridwen obtenu après changement de modèle est toujours un dataset de Spark paramétré par la classe 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 Sûretés apportées par le typage

```
dataM.as[Relation] // Ne compile pas puisque le schéma de dataM n'est pas conforme au
    modèle relationnel

data.as[Graph] // Ne compile pas puisque le schéma de data n'est pas conforme au modèle
    graphe
```

2.2 Projection d'attribut(s)

Le nouveau schéma du dataset après projection est automatiquement inféré à partir du schéma de départ des données, des noms et des types des attributs sélectionnés. Le dataset sous-jacent devient un dataframe (Dataset[Row]) puisque Spark perd la trace du schéma après application d'une projection. Le schéma du dataframe obtenu (sous forme de StructType [1]) est mis à jour pour correspondre à celui du dataset de Pridwen.

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

2.2.1 Projection simple

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

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

Model: Relation

Schema:
- att1: String

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

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

2.2.2 Projection d'un attribut imbriqué

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

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

Model: JSON

Schema:
- att2: Int

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

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

2.2.3 Projection de plusieurs attributs

```
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 Projection avec un 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 Sûretés apportées par le typage

```
data.select(col('fail)) // Ne compile pas puisque le schéma de data ne contient pas
    d'attribut nommé fail
```

2.3 Sélection de ligne(s)

Cette transformation ne modifie pas le schéma de la structure contenant les données du dataset. S'il s'agit d'un dataset de Spark paramétré par un schéma S (sous forme de classe), alors la structure interne du nouveau dataset de Pridwen obtenu après sélection est toujours un dataset de Spark paramétré par la classe S.

2.3.1 En comparant les valeurs de deux attributs

```
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 En comparant les valeurs d'un attribut avec une constante

```
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}|
+----+----+----+-------------+

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

```
// Autres opérations de comparaison prédéfinies
```

```
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)
```

Voir section B.1 pour les sorties consoles des transformations ci-dessus.

2.3.3 Avec plusieurs 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 Comparaison basée sur une UDF

```
// Appliquée sur certains attributs
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|
+----+----+-----+

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

```
// Appliquée sur tous les attributs
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 Sûretés apportées par le typage

```
dataN.filter(col('att4) === col('att7) -> col('att2)) // Ne compile pas parce que les
    attributs att4 et att7.att2 n'ont pas le même type

dataN.filter(col('att7) -> col('att1) === v(1)) // Ne compile pas parce que l'attribut
    att7.att1 n'est pas du même type que la valeur 1

```

```
data.filter(col('att1) && col('att2), (x: String, y: Int) => 0) // Ne compile pas parce
    que la valeur de retour de l'UDF n'est pas un booléen

data.filter(col('att1) && col('att2), (x: String, y: String) => true) // Ne compile pas
    parce que l'attribut att2 (Int) n'est pas de type String

data.filter(col('att1) && col('att2), (x: String, y: Int, z: Boolean) => y == 2) // Ne
    compile pas parce que l'UDF a trois paramètres d'entrée alors que seuls deux
    attributs sont sélectionnés

dataN.filter(col('fail) === col('att7) -> col('att1)) // Ne compile pas parce que
    l'attribut fail n'existe pas dans le schéma de dataN
```

2.4 Ajout d'un attribut

Le nouveau schéma du dataset après ajout de l'attribut est automatiquement inféré à partir du schéma de départ des données, du nom passé en paramètre, et de la valeur ou du type de retour de l'opérateur/de la fonction utilisée pour calculer les valeurs du nouvel attribut. Le dataset sous-jacent devient un dataframe (Dataset[Row]) puisque Spark perd la trace du schéma après ajout d'un nouvel attribut. Le schéma du dataframe obtenu (sous forme de StructType [2]) est mis à jour pour correspondre à celui du dataset de Pridwen.

2.4.1 Nouvel attribut avec des valeurs constantes

```
data.add(col('test), 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. https://spark.apache.org/docs/1.5.0/api/java/org/apache/spark/sql/types/StructType.html

```
// Ajout imbriqué
dataN.add(col('att7) -> col('test), 0).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, 0}|
|  C |  3 |true|{D, 4, false, 0}|
|  E |  5 |true|{E, 6, false, 0}|
+----+----+----+----------------+

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

```
// Ajout nécessitant un changement de modèle
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 Nouvel attribut avec des valeurs calculées à partir des valeurs d'autres attributs

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

// Autres opérations que + prédéfinies : -, *, %, /
```

```
============= 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 Nouvel attribut avec des valeurs calculées par une UDF

```
// À partir d'un seul attribut (sans imbrication)
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 |
+----+----+-----+----+

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

```
// À partir d'un seul attribut (avec imbrication)
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}|
+----+----+----+----------------+

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

```
// À partir de plusieurs attributs
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 |
+----+----+-----+----+

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

```
// À partir de tous les attributs
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 Sûretés apportées par le typage

```
data.add(col('test), List[Int]()).keepModel // Ne compile pas parce qu'un attribut
    multivalué ne peut pas être ajouté àune relation

data.add(col('test), col('att1) + v(10)).keepModel // Ne compile pas parce que
    l'attribut att1 n'est pas un attribut numérique

data.add(col('test), col('att1) + col('att2)).keepModel // Idem, seul att2 est un
    attribut numérique

data.add(col('test), col('fail) + v(10)).keepModel // Ne compile pas parce que
    l'attribut fail n'existe pas dans le schéma de data

data.add(col('test), col('att1) && col('att2), (x: String, y: String) =>
    s"$x$y").keepModel // Ne compile pas parce que la signature de la fonction passée
    en paramètre est erronée (att2 est de type Int alors que y est de type String)
```

2.5 Suppression d'un attribut

Le nouveau schéma du dataset après suppression est automatiquement inféré à partir du schéma de départ des données, duquel est retiré l'attribut dont le nom est passé en paramètre. Le dataset sous-jacent devient un dataframe (Dataset[Row]) puisque Spark perd la trace du schéma après suppression d'un attribut. Le schéma du dataframe obtenu (sous forme de StructType [3]) est mis à jour pour correspondre à celui du dataset de Pridwen.

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

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

```

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

```
Model: Relation

Schema:
- att2: Int
- att3: Boolean

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

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

2.5.1 Sûretés apportées par le typage

```
data.drop(col('fail)) // Ne compile pas parce que l'attribut fail n'existe pas dans le
    schéma de data
```

2.6 Mise à jour d'un attribut

Le nouveau schéma du dataset après mise à jour de l'attribut est automatiquement inféré à partir du schéma de départ des données, des noms d'attributs passés en paramètres et éventuellement du type de retour de la fonction utilisée pour modifier les valeurs de l'attribut. Le dataset sous-jacent devient un dataframe (Dataset[Row]) puisque Spark perd la trace du schéma après la mise à jour d'un attribut. Le schéma du dataframe obtenu (sous forme de StructType [4]) est mis à jour pour correspondre à celui du dataset de Pridwen.

2.6.1 Renommer un attribut

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

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

Model: Relation

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

+----+----+-----+
|test|att2| att3|
```

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

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

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

2.6.2 Modifier les valeurs d'un attribut

```
// Avec changement de nom
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|
+----+----+-----+

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

```
// Sans changement de nom
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|
+----+----+-----+

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

2.6.3 Sûretés apportées par le typage

```
data.update('fail, 'test, (x: String) => 0).keepModel // Ne compile pas parce que
    l'attribut fail n'existe pas dans le schéma de data

data.update('att1, 'test, (x: Int) => 0).keepModel // Ne compile pas parce que la
    signature de la fonction passée en paramètre est erronée (att1 est un attribut de
    type String alors que x est de type Int)
```

2.7 Tri d'un dataset

Cette transformation ne modifie pas le schéma de la structure contenant les données du dataset. S'il s'agit d'un dataset de Spark paramétré par un schéma S (sous forme de classe), alors la structure interne du nouveau dataset de Pridwen obtenu après tri des données est toujours un dataset de Spark paramétré par la classe S.

```
// Un critère d'ordre. Modes supporté : asc (croissant), desc (décroissant)
data.orderBy(col('att1).desc).describe(true)
```

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

Model: Relation

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

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

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

```
// Plusieurs critères d'ordre
data.orderBy(col('att1).desc && col('att2).asc).describe(true)
```

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

Model: Relation

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

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

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

2.7.1 Sûretés apportées par le typage

```
data.orderBy(col('fail).desc) // Ne compile pas parce que l'attribut fail n'existe pas
    dans le schéma de data
```

2.8 Jointure entre datasets

Le nouveau schéma du dataset après jointure est automatiquement inféré en fusionnant les schémas de départ des données. Le dataset sous-jacent devient un dataframe (Dataset[Row]) puisque Spark perd la trace du schéma après jointure. Le schéma du dataframe obtenu (sous forme de StructType [5]) est mis à jour pour correspondre à celui du dataset de Pridwen.

```
// Création d'un nouveau dataset
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

```

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

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

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

```
// Jointure interne avec une seule 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 |
+----+----+-----+----+----+

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

```
// Jointure interne avec plusieurs conditions conservant le modèle du jeu de données
    gauche (data). Supporte les mêmes opérateurs de comparaison que 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 |
+----+----+-----+----+----+

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

```
// Jointure interne conservant le modèle du jeu de données droit (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|        []     |
+----+----+-----+----+----+-----+---------------+

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

```
// Jointure interne avec changement de modèle
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|        []     |
+----+----+-----+----+----+-----+---------------+

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

```
// Avec mention explicite du mode de jointure. Modes supportés : inner, cross, outer,
    full, fullouter, full_outer, left, leftouter, left_outer, right, rightouter,
    right_outer (tout sauf semi et anti jointures, pour l'instant)
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 Sûretés apportées par le typage

```
data.join(data2, col('att1) === col('truc), "fail").keepLeftModel // Ne compile pas
    parce que "fail" n'est pas un mode de jointure supporté

```

```
data.join(dataM, col('att1) === col('att1)).keepLeftModel // Ne compile pas parce que
    le résultat serait une relation (keepLeftModel) possédant un attribut multivalué
    (att4 de dataM)

data.join(data2, col('fail) === col('truc)).keepLeftModel // Ne compile pas parce que
    l'attribut fail n'existe pas dans le schéma de data
```

2.9 Agrégation d'un dataset

Le nouveau schéma du dataset après agrégation est automatiquement inféré à partir du schéma de départ des données, des noms d'attributs passés en paramètres à la fonction groupBy et des types de retour des opérateurs passés en paramètres à la fonction agg. Les nouveaux attributs ajoutés prennent le nom de l'opérateur utilisé pour calculer leurs valeurs. Le dataset sous-jacent devient un dataframe (Dataset[Row]) puisque Spark perd la trace du schéma après agrégation. Le schéma du dataframe obtenu (sous forme de StructType [6]) est mis à jour pour correspondre à celui du dataset de Pridwen.

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

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

Model: Relation

Schema:
- att3: Boolean
- max: Int
- avg: Double
- count: Long

+-----+-----+---+---+
| att3|count|avg|max|
+-----+-----+---+---+
| true|  2  |2.0| 3 |
|false|  1  |2.0| 2 |
+-----+-----+---+---+

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

2.9.1 Sûretés apportées par le typage

```
data.groupBy(col('att3)).agg(col('att1).avg).keepModel // Ne compile pas parce que att1
    n'est pas un attribut numérique alors que l'opération avg ne s'applique que sur les
    attributs de ce type

```

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

```
3 data.groupBy(col('fail)).agg(col('fail).count).keepModel // Ne compile pas parce que
      l'attribut fail n'existe pas dans le schéma de data
```

A Exemple complet

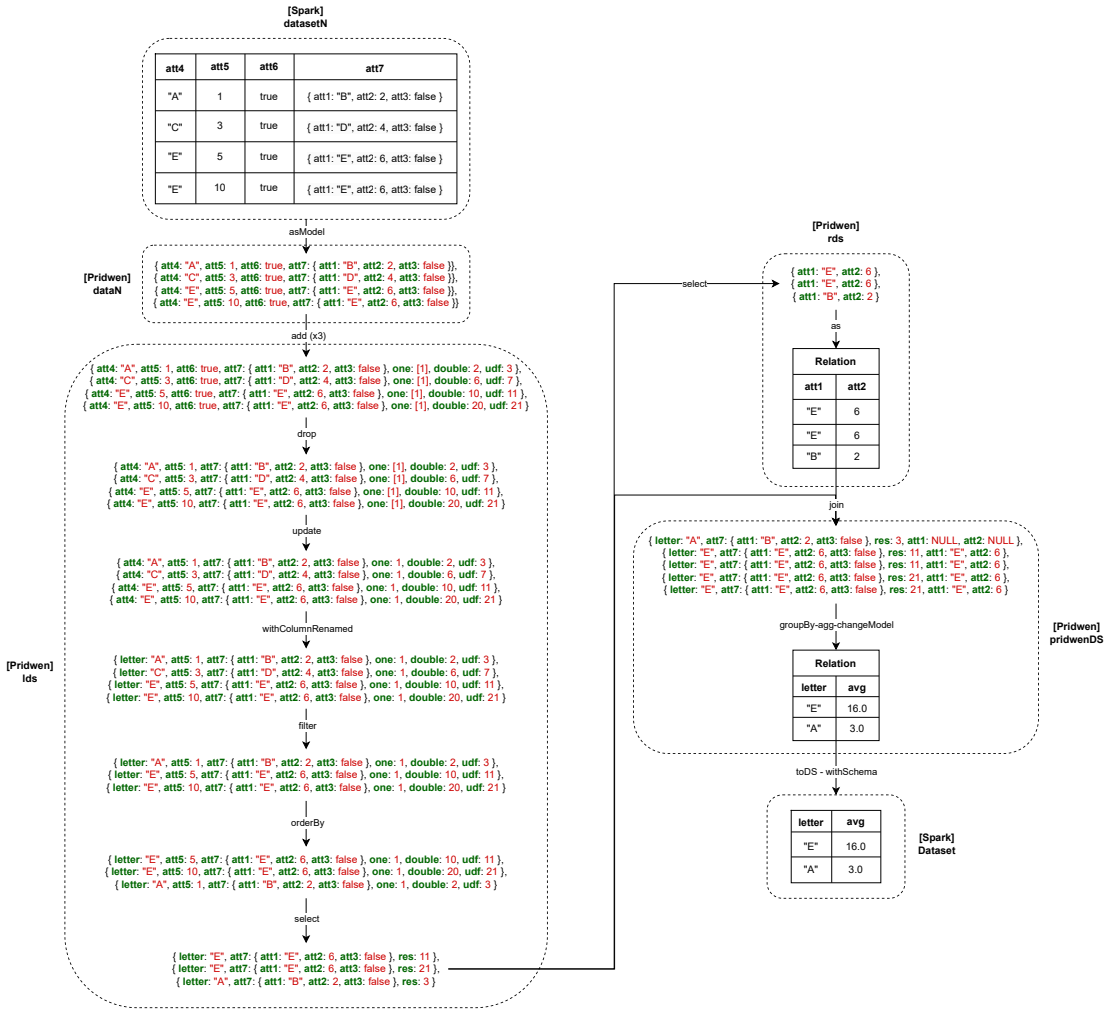

FIGURE 1 – Transformations appliquées

```
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
```

Transformations appliquées (figure 1) :

1. partir d'un dataset de Spark ;
2. le transformer en dataset de Pridwen ;
3. ajout d'un attribut (x3) ;
4. suppression d'un attribut ;
5. modifier des valeurs d'un attribut ;
6. renommer un attribut ;
7. sélection de lignes ;
8. tri d'un dataset ;
9. projection d'attributs (x2) ;
10. changement de modèle ;
11. jointure entre datasets ;
12. agrégation d'un dataset ;
13. revenir à un dataset de Spark.

```
============= 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 Autres sorties consoles

B.1 Sélection de ligne(s)

```
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|
+----+----+-----+
```

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