

# Datasets de Pridwen

Alexis Guyot

20 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

<b>1</b>	<b>Créer un dataset</b>	<b>3</b>
1.1	Partir d'un dataset de Spark	3
1.2	Le transformer en dataset de Pridwen	3
1.3	Au besoin, revenir à un dataset de Spark	5
<b>2</b>	<b>Transformer un dataset</b>	<b>6</b>
2.1	Changer de modèle	6
2.1.1	Sûretés apportées par le typage	6
2.2	Projection d'attribut(s)	6
2.2.1	Projection simple	7
2.2.2	Projection d'un attribut imbriqué	7
2.2.3	Projection de plusieurs attributs	7
2.2.4	Projection avec un alias	8
2.2.5	Sûretés apportées par le typage	9
2.3	Sélection de ligne(s)	9
2.3.1	En comparant les valeurs de deux attributs	9
2.3.2	En comparant les valeurs d'un attribut avec une constante	10
2.3.3	Avec plusieurs conditions	11
2.3.4	Comparaison basée sur une UDF	11
2.3.5	Sûretés apportées par le typage	12
2.4	Ajout d'un attribut	13
2.4.1	Nouvel attribut avec des valeurs constantes	13
2.4.2	Nouvel attribut avec des valeurs calculées à partir des valeurs d'autres attributs	15

2.4.3	Nouvel attribut avec des valeurs calculées par une UDF . . . . .	16
2.4.4	Sûretés apportées par le typage . . . . .	18
2.5	Suppression d'un attribut . . . . .	18
2.5.1	Sûretés apportées par le typage . . . . .	19
2.6	Mise à jour d'un attribut . . . . .	19
2.6.1	Renommer un attribut . . . . .	19
2.6.2	Modifier les valeurs d'un attribut . . . . .	20
2.6.3	Sûretés apportées par le typage . . . . .	21
2.7	Tri d'un dataset . . . . .	21
2.7.1	Sûretés apportées par le typage . . . . .	22
2.8	Jointure entre datasets . . . . .	22
2.8.1	Sûretés apportées par le typage . . . . .	25
2.9	Agrégation d'un dataset . . . . .	26
2.9.1	Sûretés apportées par le typage . . . . .	26
<b>A</b>	<b>Exemple complet</b>	<b>28</b>
<b>B</b>	<b>Autres sorties consoles</b>	<b>30</b>
B.1	Sélection de ligne(s) . . . . .	30

# 1 Créer un dataset

## 1.1 Partir d'un dataset de Spark

```
1 val spark = SparkSession.builder.master("local").appName("PridwenDataset").getOrCreate
2
3 import spark.implicits._
4
5 case class Base(att1: String, att2: Int, att3: Boolean)
6 case class Multivalued(att1: String, att2: Int, att3: Boolean, att4: List[Double])
7 case class Nested(att4: String, att5: Int, att6: Boolean, att7: Base)
8
9 val dataset = Seq(
10     Base("A", 1, true),
11     Base("B", 2, false),
12     Base("C", 3, true)
13 ).toDS
14
15 val datasetM = Seq(
16     Multivalued("A", 1, true, List(1.1, 1.2, 1.3)),
17     Multivalued("B", 2, false, List(2.1, 2.2)),
18     Multivalued("C", 3, true, List())
19 ).toDS
20
21 val datasetN = Seq(
22     Nested("A", 1, true, Base("B", 2, false)),
23     Nested("C", 3, true, Base("D", 4, false)),
24     Nested("E", 5, true, Base("E", 6, false))
25 ).toDS
```

## 1.2 Le transformer en dataset de Pridwen

```
1 val data = dataset.asModel[Relation]
2 val dataM = datasetM.asModel[JSON]
3 val dataN = datasetN.asModel[JSON]
4
5 // Syntaxes alternatives
6 val data = Data(dataset).as[Relation]
7 val data = Data[Relation](dataset)
```

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

```
1 ===== MyRelation
2
3 Model: Relation
4
5 Schema:
6 - att1: String
```

```

7 - att2: Int
8 - att3: Boolean
9
10 +-----+-----+-----+
11 |att1|att2| att3|
12 +-----+-----+-----+
13 |  A |  1 | true|
14 |  B |  2 |false|
15 |  C |  3 | true|
16 +-----+-----+-----+
17
18 =====

```

```

1 dataM.describe("MyMultiJSON", true)

```

```

1 ===== MyMultiJSON
2
3 Model: JSON
4
5 Schema:
6 - att1: String
7 - att2: Int
8 - att3: Boolean
9 - att4: List[Double]
10
11 +-----+-----+-----+-----+
12 |att1|att2| att3|      att4      |
13 +-----+-----+-----+-----+
14 |  A |  1 | true|[1.1, 1.2, 1.3]|
15 |  B |  2 |false|[2.1, 2.2]   |
16 |  C |  3 | true|      []      |
17 +-----+-----+-----+-----+
18
19 =====

```

```

1 dataN.describe("MyNestedJSON", true)

```

```

1 ===== MyNestedJSON
2
3 Model: JSON
4
5 Schema:
6 - att4: String
7 - att5: Int
8 - att6: Boolean
9 - att7:
10   - att1: String
11   - att2: Int

```

```

12   - att3: Boolean
13
14   +-----+-----+-----+-----+
15   |att4|att5|att6|   att7   |
16   +-----+-----+-----+-----+
17   |  A  |  1  |true|{B, 2, false}|
18   |  C  |  3  |true|{D, 4, false}|
19   |  E  |  5  |true|{E, 6, false}|
20   +-----+-----+-----+-----+
21
22   =====

```

### 1.3 Au besoin, revenir à un dataset de Spark

```

1   val sparkDS = data.toDS
2
3   // Syntaxe alternative
4   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.

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

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

#### 2.1.1 Sûretés apportées par le typage

```
1 dataM.as[Relation] // Ne compile pas puisque le schéma de dataM n'est pas conforme au
                      modèle relationnel
2
3 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<sup>1</sup>) 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

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

```
1 ===== Dataset
2
3 Model: Relation
4
5 Schema:
6 - att1: String
7
8 +----+
9 |att1|
10 +----+
11 |  A  |
12 |  B  |
13 |  C  |
14 +----+
15
16 =====
```

### 2.2.2 Projection d'un attribut imbriqué

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

```
1 ===== Dataset
2
3 Model: JSON
4
5 Schema:
6 - att2: Int
7
8 +----+
9 |att2|
10 +----+
11 |  2  |
12 |  4  |
13 |  6  |
14 +----+
15
16 =====
```

### 2.2.3 Projection de plusieurs attributs

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

```

1 ===== Dataset
2
3 Model: JSON
4
5 Schema:
6 - att4: String
7 - att2: Int
8
9 +----+----+
10 |att4|att2|
11 +----+----+
12 | A | 2 |
13 | C | 4 |
14 | E | 6 |
15 +----+----+
16
17 =====

```

#### 2.2.4 Projection avec un alias

```

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

```

```

1 ===== Dataset
2
3 Model: Relation
4
5 Schema:
6 - test: String
7
8 +----+
9 |test|
10 +----+
11 | A |
12 | B |
13 | C |
14 +----+
15
16 =====

```

```

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

```

```

1 ===== Dataset
2
3 Model: JSON
4
5 Schema:
6 - att4: String

```



```

7 - test: String
8
9 +----+----+
10 |att4|test|
11 +----+----+
12 | A | B |
13 | C | D |
14 | E | E |
15 +----+----+
16
17 =====

```

## 2.2.5 Sûretés apportées par le typage

```

1 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

```

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

```

```

1 ===== Dataset
2
3 Model: JSON
4
5 Schema:
6 - att4: String
7 - att5: Int
8 - att6: Boolean
9 - att7:
10   - att1: String
11   - att2: Int
12   - att3: Boolean
13
14 +----+----+----+-----+
15 |att4|att5|att6| att7 |
16 +----+----+----+-----+
17 | E| 5 |true|{E, 6, false}|
18 +----+----+----+-----+
19

```

```
20 =====
```

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

```
1 ===== Dataset
```

```
2
3 Model: JSON
```

```
4
5 Schema:
6 - att4: String
7 - att1: String
```

```
8
9 +---+---+
10 |att4|att1|
11 +---+---+
12 | E | E |
13 +---+---+
```

```
14
15 =====
```

### 2.3.2 En comparant les valeurs d'un attribut avec une constante

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

```
1 ===== Dataset
```

```
2
3 Model: JSON
```

```
4
5 Schema:
6 - att4: String
7 - att5: Int
8 - att6: Boolean
9 - att7:
10   - att1: String
11   - att2: Int
12   - att3: Boolean
```

```
13
14 +---+---+---+---+
15 |att4|att5|att6| att7 |
16 +---+---+---+---+
17 | C | 3 | true|{D, 4, false}|
18 +---+---+---+---+
```

```
19
20 =====
```

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

```

2 data.filter(col('att2') != v(2)).describe(true)
3 data.filter(col('att2') > v(2)).describe(true)
4 data.filter(col('att2') >= v(2)).describe(true)
5 data.filter(col('att2') < v(2)).describe(true)
6 data.filter(col('att2') <= v(2)).describe(true)
7 data.filter(col('att3').isNull()).describe(true)
8 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

```

1 dataN.filter(
2   col('att7') -> col('att1') === v("D") ||
3   col('att4') === col('att7') -> col('att1')
4 ).describe(true)

```

```

1 ===== Dataset
2
3 Model: JSON
4
5 Schema:
6 - att4: String
7 - att5: Int
8 - att6: Boolean
9 - att7:
10   - att1: String
11   - att2: Int
12   - att3: Boolean
13
14 +---+---+---+-----+
15 |att4|att5|att6| att7 |
16 +---+---+---+-----+
17 | C | 3 |true|{D, 4, false}|
18 | E | 5 |true|{E, 6, false}|
19 +---+---+---+-----+
20
21 =====

```

### 2.3.4 Comparaison basée sur une UDF

```

1 // Appliquée sur certains attributs
2 data.filter(
3   col('att1') && col('att2'),
4   (x: String, y: Int) => x == "A" || y == 2
5 ).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 |  A  |  1  | true|
14 |  B  |  2  |false|
15 +----+----+----+
16
17 =====

```

```

1 // Appliquée sur tous les attributs
2 data.filter((x: String, y: Int, z: Boolean) => x == "A" || y == 2 || z).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 |  A  |  1  | true|
14 |  B  |  2  |false|
15 |  C  |  3  | true|
16 +----+----+----+
17
18 =====

```

### 2.3.5 Sûretés apportées par le typage

```

1 dataN.filter(col('att4) == col('att7) -> col('att2)) // Ne compile pas parce que les
2   attributs att4 et att7.att2 n'ont pas le même type
3
4 dataN.filter(col('att7) -> col('att1) == v(1)) // Ne compile pas parce que l'attribut
5   att7.att1 n'est pas du même type que la valeur 1

```

```

5 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
6
7 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
8
9 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
10
11 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<sup>2</sup>) est mis à jour pour correspondre à celui du dataset de Pridwen.

### 2.4.1 Nouvel attribut avec des valeurs constantes

```

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 =====

```

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

```

1 // Ajout imbriqué
2 dataN.add(col('att7') -> col('test'), 0).keepModel.describe(true)

```

```

1 ===== Dataset
2
3 Model: JSON
4
5 Schema:
6 - att4: String
7 - att5: Int
8 - att6: Boolean
9 - att7:
10   - att1: String
11   - att2: Int
12   - att3: Boolean
13   - test: Int
14
15 +---+---+---+---+
16 |att4|att5|att6|   att7   |
17 +---+---+---+---+
18 |  A  |  1  |true|{B, 2, false, 0}|
19 |  C  |  3  |true|{D, 4, false, 0}|
20 |  E  |  5  |true|{E, 6, false, 0}|
21 +---+---+---+---+
22
23 =====

```

```

1 // Ajout nécessitant un changement de modèle
2 data.add(col('test'), List[Int](0)).changeModel[JSON].describe(true)

```

```

1 ===== Dataset
2
3 Model: JSON
4
5 Schema:
6 - att1: String
7 - att2: Int
8 - att3: Boolean
9 - test: List[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 =====

```

## 2.4.2 Nouvel attribut avec des valeurs calculées à partir des valeurs d'autres attributs

```
1 data.add(col('test'), col('att2') + v(10)).keepModel.describe(true)
2
3 // Autres opérations que + prédéfinies : -, *, %, /
```

```
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| 11 |
15 |  B |  2 |false| 12 |
16 |  C |  3 | true| 13 |
17 +---+---+---+---+
18
19 =====
```

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

```
1 ===== Dataset
2
3 Model: JSON
4
5 Schema:
6 - att4: String
7 - att5: Int
8 - att6: Boolean
9 - att7:
10   - att1: String
11   - att2: Int
12   - att3: Boolean
13 - test: Int
14
15 +---+---+---+-----+---+
16 |att4|att5|att6| att7      |test|
17 +---+---+---+-----+---+
18 |  A |  1 | true|{B, 2, false}| 3 |
```

```

19 | C | 3 | true | {D, 4, false} | 7 |
20 | E | 5 | true | {E, 6, false} | 11 |
21 +-----+-----+-----+-----+
22
23 =====

```

### 2.4.3 Nouvel attribut avec des valeurs calculées par une UDF

```

1 // À partir d'un seul attribut (sans imbrication)
2 data.add(col('test'), col('att2'), (x: Int) => x + 1).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 | 2 |
15 | B | 2 | false | 3 |
16 | C | 3 | true | 4 |
17 +-----+-----+-----+-----+
18
19 =====

```

```

1 // À partir d'un seul attribut (avec imbrication)
2 dataN.add(
3   col('att7') -> col('test'),
4   col('att7') -> col('att2'),
5   (x: Int) => x + 1
6 ).keepModel.describe(true)

```

```

1 ===== Dataset
2
3 Model: JSON
4
5 Schema:
6 - att4: String
7 - att5: Int
8 - att6: Boolean
9 - att7:

```



```

10   - att1: String
11   - att2: Int
12   - att3: Boolean
13   - test: Int
14
15   +-----+-----+-----+-----+
16   |att4|att5|att6|      att7      |
17   +-----+-----+-----+-----+
18   |  A  |  1  |true|{B, 2, false, 3}|
19   |  C  |  3  |true|{D, 4, false, 5}|
20   |  E  |  5  |true|{E, 6, false, 7}|
21   +-----+-----+-----+-----+
22
23   =====

```

```

1  // À partir de plusieurs attributs
2  data.add(
3    col('test), col('att1) && col('att2),
4    (x: String, y: Int) => s"$x$y"
5  ).keepModel.describe(true)

```

```

1  ===== Dataset
2
3  Model: Relation
4
5  Schema:
6  - att1: String
7  - att2: Int
8  - att3: Boolean
9  - test: String
10
11  +-----+-----+-----+-----+
12  |att1|att2| att3|test|
13  +-----+-----+-----+-----+
14  |  A  |  1  | true| A1 |
15  |  B  |  2  |false| B2 |
16  |  C  |  3  | true| C3 |
17  +-----+-----+-----+-----+
18
19  =====

```

```

1  // À partir de tous les attributs
2  data.add(col('test), "*", (x: String, y: Int, z: Boolean) => 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 =====

```

#### 2.4.4 Sûretés apportées par le typage

```

1 data.add(col('test'), List[Int]()).keepModel // Ne compile pas parce qu'un attribut
      multivalué ne peut pas être ajouté à une relation
2
3 data.add(col('test'), col('att1') + v(10)).keepModel // Ne compile pas parce que
      l'attribut att1 n'est pas un attribut numérique
4
5 data.add(col('test'), col('att1') + col('att2')).keepModel // Idem, seul att2 est un
      attribut numérique
6
7 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
8
9 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<sup>3</sup>) est mis à jour pour correspondre à celui du dataset de Pridwen.

```

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

```

```

1 ===== Dataset
2

```

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

```

3 Model: Relation
4
5 Schema:
6 - att2: Int
7 - att3: Boolean
8
9 +-----+
10 |att2| att3|
11 +-----+
12 |  1 | true |
13 |  2 | false|
14 |  3 | true |
15 +-----+
16
17 =====

```

### 2.5.1 Sûretés apportées par le typage

```

1 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<sup>4</sup>) est mis à jour pour correspondre à celui du dataset de Pridwen.

### 2.6.1 Renommer un attribut

```

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

```

```

1 ===== Dataset
2
3 Model: Relation
4
5 Schema:
6 - test: String
7 - att2: Int
8 - att3: Boolean
9
10 +-----+-----+
11 |test|att2| att3|

```

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

```

12 +-----+
13 | A | 1 | true |
14 | B | 2 | false|
15 | C | 3 | true |
16 +-----+
17
18 =====

```

## 2.6.2 Modifier les valeurs d'un attribut

```

1 // Avec changement de nom
2 data.update('att1', 'test', (x: String) => 0).keepModel.describe(true)

```

```

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

```

```

1 // Sans changement de nom
2 data.update('att1', (x: String) => 0).keepModel.describe(true)

```

```

1 ===== Dataset
2
3 Model: Relation
4
5 Schema:
6 - att1: Int
7 - att2: Int
8 - att3: Boolean
9
10 +-----+
11 |att1|att2| att3|
12 +-----+

```

```

13 | 0 | 1 | true|
14 | 0 | 2 | false|
15 | 0 | 3 | true|
16 +---+---+---+
17
18 =====

```

### 2.6.3 Sûretés apportées par le typage

```

1 data.update('fail', 'test', (x: String) => 0).keepModel // Ne compile pas parce que
  l'attribut fail n'existe pas dans le schéma de data
2
3 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.

```

1 // Un critère d'ordre. Modes supporté : asc (croissant), desc (décroissant)
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 // Plusieurs critères d'ordre
2 data.orderBy(col('att1').desc && col('att2').asc).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 =====

```

### 2.7.1 Sûretés apportées par le typage

```

1 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<sup>5</sup>) est mis à jour pour correspondre à celui du dataset de Pridwen.

```

1 // Création d'un nouveau dataset
2 val data2 = data.drop(col('att3'))
3               .add(col('test'), col('att2') % v(2))
4               .keepModel.drop(col('att2'))
5               .withColumnRenamed('att1', 'truc')
6 data2.describe(true) // Relation

```

```

1 ===== Dataset
2
3 Model: Relation
4
5 Schema:
6 - truc: String
7 - test: Int
8

```

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

```

9  +-----+-----+
10 |truc|test|
11 +-----+-----+
12 |  A |  1 |
13 |  B |  0 |
14 |  C |  1 |
15 +-----+-----+
16
17 =====

```

```

1  // Jointure interne avec une seule condition
2  data.join(data2, col('att1') === col('truc')).keepLeftModel.describe(true)

```

```

1  ===== Dataset
2
3  Model: Relation
4
5  Schema:
6  - att1: String
7  - att2: Int
8  - att3: Boolean
9  - truc: String
10 - test: Int
11
12 +-----+-----+-----+-----+-----+
13 |att1|att2| att3|truc|test|
14 +-----+-----+-----+-----+-----+
15 |  A |  1 | true| A |  1 |
16 |  B |  2 |false| B |  0 |
17 |  C |  3 | true| C |  1 |
18 +-----+-----+-----+-----+-----+
19
20 =====

```

```

1  // Jointure interne avec plusieurs conditions conservant le modèle du jeu de données
   gauche (data). Supporte les mêmes opérateurs de comparaison que filter.
2  data.join(
3      data2,
4      col('att1') === col('truc') && col('att2') > col('test')
5  ).keepLeftModel.describe(true)

```

```

1  ===== Dataset
2
3  Model: Relation
4
5  Schema:
6  - att1: String
7  - att2: Int

```

```

8 - att3: Boolean
9 - truc: String
10 - test: Int

```

```

11
12 +---+---+---+---+---+
13 |att1|att2| att3|truc|test|
14 +---+---+---+---+---+
15 |  B |  2 |false|  B |  0 |
16 |  C |  3 | true|  C |  1 |
17 +---+---+---+---+---+

```

```

18 =====
19

```

```

1 // Jointure interne conservant le modèle du jeu de données droit (dataM)
2 data.join(dataM, col('att1') === col('att1')).keepRightModel.describe(true)

```

```

1 ===== Dataset

```

```

2
3 Model: JSON

```

```

4
5 Schema:

```

```

6 - att1: String
7 - att2: Int
8 - att3: Boolean
9 - att1: String
10 - att2: Int
11 - att3: Boolean
12 - att4: List[Double]

```

```

13
14 +---+---+---+---+---+---+---+
15 |att1|att2| att3|att1|att2| att3| att4          |
16 +---+---+---+---+---+---+---+
17 |  A |  1 | true|  A |  1 | true|[1.1, 1.2, 1.3]|
18 |  B |  2 |false|  B |  2 |false|[2.1, 2.2]  |
19 |  C |  3 | true|  C |  3 | true|          []    |
20 +---+---+---+---+---+---+---+

```

```

21 =====
22

```

```

1 // Jointure interne avec changement de modèle
2 data.join(dataM, col('att1') === col('att1')).changeModel[JSON].describe(true)

```

```

1 ===== Dataset

```

```

2
3 Model: JSON

```

```

4
5 Schema:

```

```

6 - att1: String

```



```

7 - att2: Int
8 - att3: Boolean
9 - att1: String
10 - att2: Int
11 - att3: Boolean
12 - att4: List[Double]
13
14 +-----+-----+-----+-----+-----+-----+-----+-----+
15 |att1|att2| att3|att1|att2| att3| att4          |
16 +-----+-----+-----+-----+-----+-----+-----+-----+
17 |  A |  1 | true|  A |  1 | true|[1.1, 1.2, 1.3]|
18 |  B |  2 |false|  B |  2 |false| [2.1, 2.2]  |
19 |  C |  3 | true|  C |  3 | true|          []   |
20 +-----+-----+-----+-----+-----+-----+-----+-----+
21
22 =====

```

```

1 // 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)
2 data.join(data2, col('att1) === col('truc), "left").keepLeftModel.describe(true)

```

```

1 ===== Dataset
2
3 Model: Relation
4
5 Schema:
6 - att1: String
7 - att2: Int
8 - att3: Boolean
9 - truc: String
10 - test: Int
11
12 +-----+-----+-----+-----+-----+
13 |att1|att2| att3|truc|test|
14 +-----+-----+-----+-----+-----+
15 |  A |  1 | true|  A |  1 |
16 |  B |  2 |false|  B |  0 |
17 |  C |  3 | true|  C |  1 |
18 +-----+-----+-----+-----+-----+
19
20 =====

```

### 2.8.1 Sûretés apportées par le typage

```

1 data.join(data2, col('att1) === col('truc), "fail").keepLeftModel // Ne compile pas
  parce que "fail" n'est pas un mode de jointure supporté
2

```

```

3 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)
4
5 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`<sup>6</sup>) est mis à jour pour correspondre à celui du dataset de Pridwen.

```

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 Sûretés apportées par le typage

```

1 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
2

```

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

```
1 val datasetN = Seq(  
2   SchemaDepartNested("A", 1, true, SchemaDepart("B", 2, false)),  
3   SchemaDepartNested("C", 3, true, SchemaDepart("D", 4, false)),  
4   SchemaDepartNested("E", 5, true, SchemaDepart("E", 6, false)),  
5   SchemaDepartNested("E", 10, true, SchemaDepart("E", 6, false))  
6 ).toDS  
7  
8 val dataN = datasetN.asModel[JSON]  
9  
10 val lds = dataN.add(col('one), List[Int](1)).keepModel  
11   .add(col('double), col('att5) * v(2)).keepModel  
12   .add(col('udf), col('double) && col('one) && col('att6), (x: Int, y: List[Int], z:  
13     Boolean) => if(z) x + y(0) else x - y(0)).keepModel  
14   .drop(col('att6))  
15   .update('one, (x: List[Int]) => x(0)).keepModel  
16   .withColumnRenamed('att4, 'letter)  
17   .filter(col('letter) === col('att7) -> col('att1) || col('udf) === v(3))  
18   .orderBy(col('letter).desc)  
19   .select(col('letter) && col('att7) && col('udf).as('res))  
20 lds.describe(true)  
21  
22 val rds = lds.select(col('att7) -> col('att1) && col('att7) -> col('att2)).as[Relation]  
23 rds.describe(true)  
24  
25 val pridwenDS = lds.join(rds, col('letter) === col('att1), "left").keepLeftModel  
26   .groupBy(col('letter))  
27   .agg(col('res).avg)  
28   .changeModel[Relation]  
29 pridwenDS.describe(true)  
30  
31 case class FinalSchema(letter: String, avg: Double)  
pridwenDS.toDS.withSchema[FinalSchema].show
```

### Transformations appliquées :

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.

```

1  ===== Dataset
2
3  Model: JSON
4
5  Schema:
6  - letter: String
7  - att7:
8      - att1: String
9      - att2: Int
10     - att3: Boolean
11  - res: Int
12
13  +-----+-----+-----+
14  |letter|    att7    |res|
15  +-----+-----+-----+
16  |  E  |{E, 6, false}| 11|
17  |  E  |{E, 6, false}| 21|
18  |  A  |{B, 2, false}|  3|
19  +-----+-----+-----+
20
21  =====
22
23  ===== Dataset
24
25  Model: Relation
26
27  Schema:
28  - att1: String
29  - att2: Int
30
31  +----+----+
32  |att1|att2|
33  +----+----+
34  |  E  |  6  |
35  |  E  |  6  |
36  |  B  |  2  |
37  +----+----+
38
39  =====
40
41  ===== Dataset
42
43  Model: Relation
44
45  Schema:
46  - letter: String
47  - avg: Double
48
49  +-----+-----+

```

```

50 |letter| avg|
51 +-----+-----+
52 |    E |16.0|
53 |    A | 3.0|
54 +-----+-----+
55
56 =====
57
58 +-----+-----+
59 |letter| avg|
60 +-----+-----+
61 |    E |16.0|
62 |    A | 3.0|
63 +-----+-----+

```

## B Autres sorties consoles

### B.1 Sélection de ligne(s)

```

1 data.filter(col('att2') != v(2)).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 |  A |  1 |true|
14 |  C |  3 |true|
15 +----+-----+-----+
16
17 =====

```

```

1 data.filter(col('att2') > v(2)).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 +-----+-----+-----+

```

```

15 =====
16

```

```

1 data.filter(col('att2') >= v(2)).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 |  B  |  2  |false|
14 |  C  |  3  | true|
15 +-----+-----+-----+

```

```

16 =====
17

```

```

1 data.filter(col('att2') < v(2)).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 |  A  |  1  |true|
14 +-----+-----+-----+

```

```

15

```

```
16 =====
```

```
1 data.filter(col('att2') <= v(2)).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 |  A |  1 | true|
14 |  B |  2 |false|
15 +----+----+----+
16 =====
17
```

```
1 data.filter(col('att3').isNull()).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 +----+----+----+
14
15 =====
```

```
1 data.filter(col('att3').isNotNull()).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 |  A  |  1  | true|
14 |  B  |  2  |false|
15 |  C  |  3  | true|
16 +-----+-----+-----+
17
18 =====
```