Développement avec Spark

BIOINFO@LIPME

Spark: The Definitive Guide: Big Data Processing Made Simple

Bill Chambers & Matei Zaharia

O'REILLY

> 15j à 100%

~ 3h/j pendant 6 semaines

➔ en 2,5h on va se focaliser sur les fondamentaux du développement Spark

Les fondamentaux spécifiques de la programmation Spark

- **Données distribuées**
  - DataFrame // DataSet // RDD
    - schema
      - “A DataFrame is the most common Structured API and simply represents a table of data with rows and columns. The list that defines the columns and the types within those columns is called the *schema*. You can think of a DataFrame as a spreadsheet with named columns.”
    - Partitions
- **Données immutables/immuables**
- **Lazy evaluation**
  - transformations et actions
  - « Execution plan // DAG // scheduler »
- **Collecte des données en mémoire du « driver »**

Illustration à partir d'une analyse biblio

- **Source de données: PUBMED**
- **1114 fichiers XML compressés téléchargés compressés (35Go)**

  - Petite analyse des publications INRAE en se basant sur les affiliations des 1er et dernier auteurs.
  - « Labelliser » les publications qui sont assurément du domaine végétal et du domaine animal

spark-shell

Ne remplace pas un IDE tel qu'IntelliJ IDEA mais très pratique pour apprendre/tester

```
% sudo update-java-alternatives -l
% sudo update-java-alternatives -s /usr/lib/jvm/java-1.8.0-openjdk-amd64   ← "blocages" en v11
% wget https://dlcdn.apache.org/spark/spark-3.3.1/spark-3.3.1-bin-hadoop3.tgz
% gzip -cd spark-3.3.1-bin-hadoop3.tgz | tar xf -
% scala -version
# Scala code runner version 2.12.15 -- Copyright 2002-2021, LAMP/EPFL and Lightbend, Inc.
# version 2.12, trouver sur le dépôt la version du package spark-xml disponible: ici 0.15.0
# https://mvnrepository.com/artifact/com.databricks/spark-xml_2.12
% spark-3.3.1-bin-hadoop3/bin/spark-shell --packages com.databricks:spark-xml_2.12:0.15.0
Spark context Web UI available at http://192.168.1.14:4043   ← l'url web pour la supervision
Spark context available as 'sc' (master = local[*], app id = local-1667916118482).
                                          ^-- tous les cœurs, --master 'local[2]' pour 2 coeurs
Spark session available as 'spark'        ← l'objet session de spark

scala> spark.<TAB>  ← ensemble des méthodes applicables
baseRelationToDataFrame   conf  emptyDataFrame   experimental  newSession
readStream      sparkContext   stop      time      catalog  createDataFrame
emptyDataset     implicits    range   sessionState   sql streams   udf
close  createDataset     executeCommand   listenerManager   read
sharedState    sqlContext     table     version
```

spark-shell

```
scala>:help
All commands can be abbreviated, e.g., :he instead of :help.
:edit <id>|<line>        edit history
:help [command]          print this summary or command-specific help
:history [num]           show the history (optional num is commands to show) +
:h? <string>             search the history
:load <path>             interpret lines in a file
:paste [-raw] [path]     enter paste mode or paste a file
:quit                    exit the interpreter
:require <path>          add a jar to the classpath
:save <path>             save replayable session to a file
:sh <command line>       run a shell command (result is implicitly => List[String])
```

```
spark-shell -i fichier.scala : exécuter un fichier/programme
spark-shell --driver-memory 4G (1G par défaut)
```

Lecture xml avec inférence automatique du schema

types simples (string, long) et complex (struct, array)

```
scala> import com.databricks.spark.xml._
scala> val xmlfiles = "/home/atelier/Atelier/pubmed22/pubmed22n1000.xml.gz"  ← on peut mettre des '*'
scala> val art = spark.read.format("xml").option("rowTag","Article").option("excludeAttribute",
    true).option("compression","gzip").load(xmlfiles)
art: org.apache.spark.sql.DataFrame = [Abstract: struct<AbstractText: array<string>, CopyrightInformation:
    string>, ArticleDate: struct<Day: bigint, Month: bigint ... 1 more field> ... 10 more fields]

scala> art.printSchema(4) ← on limite a 4 la profondeur pour l'affichage
root
 |-- Abstract: struct (nullable = true)
 |    |-- AbstractText: array (nullable = true)
 |    |    |-- element: string (containsNull = true)
 |    |-- CopyrightInformation: string (nullable = true)
 |-- ArticleDate: struct (nullable = true)
 |    |-- Day: long (nullable = true)
 |    |-- Month: long (nullable = true)
 |    |-- Year: long (nullable = true)                          <-----
 |-- ArticleTitle: string (nullable = true)                     <-----
 |-- AuthorList: struct (nullable = true)
 |    |-- Author: array (nullable = true)                       <-----
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- AffiliationInfo: array (nullable = true)    <-----
 |    |    |    |-- CollectiveName: string (nullable = true)
 |    |    |    |-- ForeName: string (nullable = true)
 |    |    |    |-- Identifier: string (nullable = true)
 |    |    |    |-- Initials: string (nullable = true)
 |    |    |    |-- LastName: string (nullable = true)
```

.select des éléments pertinents dans un nouveau DataFrame

```
val artmin =art.select("ArticleTitle","Journal.Title","Journal.JournalIssue.PubDate.Year","AuthorList.Author") \\
              .repartition(5) ← réparti les données en 5 partitions
scala> artmin.show(3,30)
+------------------------------+------------------------------+----+------------------------------+
|                  ArticleTitle|                         Title|Year|                        Author|
+------------------------------+------------------------------+----+------------------------------+
|Well water testing in Afric...|The Science of the total en...|2019|[{[{Department of Environme...|
|Range of Normal Serum Amino...|   Transplantation proceedings|null|[{[{Division of Gastroenter...|
|A Novel Mutation in the Glu...|American journal of hyperte...|2019|[{[{Department of Cardiolog...|
+------------------------------+------------------------------+----+------------------------------+
scala> artmin.printSchema()
root
 |-- ArticleTitle: string (nullable = true)
 |-- Title: string (nullable = true)
 |-- Year: long (nullable = true)
 |-- Author: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- AffiliationInfo: array (nullable = true)
 |    |    |    |-- element: struct (containsNull = true)
 |    |    |    |    |-- Affiliation: string (nullable = true)
 |    |    |    |    |-- Identifier: array (nullable = true)
 |    |    |    |    |    |-- element: string (containsNull = true)
 |    |    |-- CollectiveName: string (nullable = true)
 |    |    |-- ForeName: string (nullable = true)
 |    |    |-- Identifier: string (nullable = true)
 |    |    |-- Initials: string (nullable = true)
 |    |    |-- LastName: string (nullable = true)
 |    |    |-- Suffix: string (nullable = true)
```

.select : transformation

.show : action

Extraction/calcul de nouvelles "features"

```
val artft = artmin.withColumn("NumAuth",size($"Author"))          ← taille de l'array
                  .withColumn("First",element_at($"Author",1))    ← extraction du premier élément
                  .withColumn("Last",element_at($"Author",-1))    ← extraction du dernier élément
                  .withColumn("Part",spark_partition_id) ← rajout de l'id de partition pour suivre partitionnement
artft.limit(5).show()
```

```
+--------------------+--------------------+----+--------------------+-------+--------------------+--------------------+----+
|        ArticleTitle|               Title|Year|              Author|NumAuth|               First|                Last|Part|
+--------------------+--------------------+----+--------------------+-------+--------------------+--------------------+----+
|Primary Leiomyosa...|Journal of gastro...|2020|[{[{Department of...|      5|{[{Department of ...|{[{Department of ...|   0|
|N-acetylcysteine ...|American journal ...|2019|[{[{Department of...|      6|{[{Department of ...|{[{Department of ...|   0|
|Sleep quality ass...|Brazilian journal...|null|[{[{Universidade ...|      5|{[{Universidade d...|{[{Universidade d...|   0|
|[Saliva as highly...|Klinicheskaia lab...|2019|[{[{Lobachevsky S...|      2|{[{Lobachevsky St...|{[{Lobachevsky St...|   0|
|Biochemical study...|Journal of food b...|2019|                null|     -1|                null|                null|   0|
+--------------------+--------------------+----+--------------------+-------+--------------------+--------------------+----+
```

```
artft.select($"First").printSchema
root
 |-- First: struct (nullable = true)
 |    |-- AffiliationInfo: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- Affiliation: string (nullable = true)
 |    |    |    |-- Identifier: array (nullable = true)
 |    |    |    |    |-- element: string (containsNull = true)
 |    |-- CollectiveName: string (nullable = true)
 |    |-- ForeName: string (nullable = true)
 |    |-- Identifier: string (nullable = true)
 |    |-- Initials: string (nullable = true)
 |    |-- LastName: string (nullable = true)
 |    |-- Suffix: string (nullable = true)
```

plusieurs façons de nommer les colonnes

```
$"Author"
col("Author")
'Author
```

plus une en SQL

```
Author
```

Extraction des affiliations et traitement des données manquantes

```
val artftaff = artft.withColumn("AffFirst",$"First".getField("AffiliationInfo").getField("Affiliation")(0))
                    .withColumn("AffLast",$"Last".getField("AffiliationInfo").getField("Affiliation")(0))
                    .na.drop()     ← le package .na permet de traiter les données manquantes

artftaff.select($"Title",$"NumAuth",$"AffFirst",$"AffLast").show(3)
+--------------------+-------+--------------------+--------------------+
|               Title|NumAuth|            AffFirst|             AffLast|
+--------------------+-------+--------------------+--------------------+
|Journal of gastro...|      5|[Department of Ge...|[Department of Ge...|
|American journal ...|      6|[Department of An...|[Department of An...|
|Klinicheskaia lab...|      2|[Lobachevsky Stat...|[Lobachevsky Stat...|
+--------------------+-------+--------------------+--------------------+

artftaff.summary().show()  ← très utile pour voir si les données manquantes et les reformatage ont été traité
                             types string et long mais pas les array, struct
                             .describe(): idem
+-------+--------------------+--------------------+------------------+-----------------+------------------+
|summary|        ArticleTitle|               Title|              Year|          NumAuth|              Part|
+-------+--------------------+--------------------+------------------+-----------------+------------------+
|  count|               19690|               19690|             19690|            19690|             19690|
|   mean|                null|                null|2019.1422549517522|5.865312341289995|1.9957846622651092|
| stddev|                null|                null| 0.566764677096882|5.645614671809945| 1.409800955332618|
|    min|                    |2018 Internationa...|              1982|                1|                 0|
|    25%|                null|                null|              2019|                3|                 1|
|    50%|                null|                null|              2019|                5|                 2|
|    75%|                null|                null|              2019|                7|                 3|
|    max|Minimal-activity... |            mSystems|              2021|              472|                 4|
+-------+--------------------+--------------------+------------------+-----------------+------------------+
```

.na.replace("ArticleTitle",Map("" -> null)).na.drop()

Extraction des publications INRA

```
val isINRA = $"AffFirst".contains("INRA").or($"AffLast".contains("INRA")) ← isINRA est de type Column

val artinrae = artftaff.where(isINRA).drop($"First").drop($"Last").drop($"Author")  ← élimination colonnes

artinrae.show(3)
+--------------------+--------------------+----+-------+----+--------------------+--------------------+
|        ArticleTitle|               Title|Year|NumAuth|Part|            AffFirst|             AffLast|
+--------------------+--------------------+----+-------+----+--------------------+--------------------+
|Construction of a...|  Scientific reports|2019|      7|   0|Toulouse Biotechn...|Toulouse Biotechn...|
|Alterations of HD...|Journal of physio...|2019|      9|   0|Service de rhumat...|Unité de Nutritio...|
|Co-circulation an...|Transboundary and...|2019|     12|   0|Institut Pasteur,...|ASTRE, Univ Montp...|
+--------------------+--------------------+----+-------+----+--------------------+--------------------+
val artinrae5 = artinrae.where("NumAuth > 5")

artinrae5.explain("formatted")  ← affichage du plan d'exécution
```

Plan d'exécution .. qui ne suit pas la séquence du code

```
(1) Scan
      XmlRelation(com.databricks.spark.xml.DefaultSource$$Lambda$2147/1077297779@1b0cefc8,Some(/home/gouzy/Atelier/pu
      bmed22/pubmed22n1000.xml.gz),Map(rowtag -> Article, excludeattribute -> true, compression -> gzip, path ->
      /home/gouzy/Atelier/pubmed22/pubmed22n1000.xml.gz),null)
Output [3]: [ArticleTitle#2, Journal#7, AuthorList#3]
ReadSchema:
      struct<ArticleTitle:string,Journal:struct<ISOAbbreviation:string,ISSN:string,JournalIssue:struct<Issue:string,P
      ubDate:struct<Day:bigint,MedlineDate:string,Month:string,Season:string,Year:bigint>,Volume:string>,Title:string
(2) Project
Output [4]: [ArticleTitle#2, Journal#7.Title AS Title#24, Journal#7.JournalIssue.PubDate.Year AS Year#25L,
      AuthorList#3.Author AS Author#26]
Input [3]: [ArticleTitle#2, Journal#7, AuthorList#3]
(3) Exchange
Input [4]: [ArticleTitle#2, Title#24, Year#25L, Author#26]
Arguments: RoundRobinPartitioning(5), REPARTITION_BY_NUM, [plan_id=552]
(4) Project
Output [8]: [ArticleTitle#2, Title#24, Year#25L, Author#26, size(Author#26, true) AS NumAuth#51,
      element_at(Author#26, 1, None, false) AS First#57, element_at(Author#26, -1, None, false) AS Last#64,
      SPARK_PARTITION_ID() AS Part#72]
Input [4]: [ArticleTitle#2, Title#24, Year#25L, Author#26]
(5) Filter
Input [8]: [ArticleTitle#2, Title#24, Year#25L, Author#26, NumAuth#51, First#57, Last#64, Part#72]
Condition : (atleastnnonnulls(10, ArticleTitle#2, Title#24, Year#25L, Author#26, NumAuth#51, First#57, Last#64,
      Part#72, First#57.AffiliationInfo.Affiliation[0], Last#64.AffiliationInfo.Affiliation[0]) AND
      ((Contains(First#57.AffiliationInfo.Affiliation[0], INRA) OR Contains(Last#64.AffiliationInfo.Affiliation[0],
      INRA)) AND (NumAuth#51 > 5)))   ← tous les filtres sont appliqués en même temps
(6) Project
Output [7]: [ArticleTitle#2, Title#24, Year#25L, NumAuth#51, Part#72, First#57.AffiliationInfo.Affiliation[0] AS
      AffFirst#118, Last#64.AffiliationInfo.Affiliation[0] AS AffLast#128]
Input [8]: [ArticleTitle#2, Title#24, Year#25L, Author#26, NumAuth#51, First#57, Last#64, Part#72]
(7) AdaptiveSparkPlan
Output [7]: [ArticleTitle#2, Title#24, Year#25L, NumAuth#51, Part#72, AffFirst#118, AffLast#128]
```

Attention aux lectures et relectures des données

```
val isINRA = $"AffFirst".contains("INRA").or($"AffLast".contains("INRA"))

val artinrae = artftaff.where(isINRA).drop($"First").drop($"Last").drop($"Author")

artinrae.show(3) ← c'est une action, le fichier est lu

....

val artinrae5 = artinrae.where("NumAuth > 5")

artinrae5.explain("formatted")  ← affichage du plan d'exécution
artinrae5.show(3) ← c'est une action et d'après le plan que l'on vient de voir le fichier est relu
```

Pour éviter que le fichier soit lu 2 fois, on peut mettre en cache le DataFrame

```
...
val artinrae = artftaff.where(isINRA).drop($"First").drop($"Last").drop($"Author").persist
artinrae.show(3)
val artinrae5 = artinrae.where("NumAuth > 5")
artinrae5.show(3)
```

➔ **En production, il faudra essayer d'éviter les actions sur les DataFrame "intermédiaires" car cela bloque la définition d'un Plan d'exécution optimal.**

User Defined Function (UDF) - extension de Spark

```
def isFrom(first: String, last: String, a_lst: List[String]): Boolean = {
    for (str <- a_lst) {
       if (first.contains(str) || last.contains(str)) {return true};
    }
    return false
}
```

← écriture en scala d'une fonction qui teste la présence d'une des chaines

```
val isFromUDF = udf[Boolean,String,String,List[String]](isFrom)
```

← enregistrement de la fonction

```
val inraeLabels = List("INRA","CEMAGREF","IRSTEA")
// les UDF prennent en paramètre des colonnes
val artinraeudf = artftaff.withColumn("isFromLabels",typedlit(inraeLabels))
                          .withColumn("isFrom",isFromUDF($"AffFirst",$"AffLast",$"isFromLabels"))
                          .where($"isFrom")
artinraeudf.select($"Title",$"AffFirst",$"AffLast",$"isFromLabels",$"isFrom").show(5)
```

← nouvelle colonne "constante"

```
+--------------------+--------------------+--------------------+--------------------+------+
|               Title|            AffFirst|             AffLast|        isFromLabels|isFrom|
+--------------------+--------------------+--------------------+--------------------+------+
|  Scientific reports|Toulouse Biotechn...|Toulouse Biotechn...|[INRA, CEMAGREF, ...|  true|
|Journal of physio...|Service de rhumat...|Unité de Nutritio...|[INRA, CEMAGREF, ...|  true|
|Transboundary and...|Institut Pasteur,...|ASTRE, Univ Montp...|[INRA, CEMAGREF, ...|  true|
|      Food chemistry|UMR STLO, Agrocam...|UMR STLO, Agrocam...|[INRA, CEMAGREF, ...|  true|
|Applied microbiol...|UMR SPO, INRA, Mo...|UMR SPO, INRA, Mo...|[INRA, CEMAGREF, ...|  true|
+--------------------+--------------------+--------------------+--------------------+------+
```

```
val mpiLabels = List("MPI","Max Planck Institute")
val artmpiudf = artftaff.withColumn("isFromLabels", typedlit(mpiLabels))
                        .withColumn("isFrom", isFromUDF($"AffFirst", $"AffLast", $"isFromLabels"))
                        .where($"isFrom")
```

Tous les types de jointures sont disponibles (y compris cross)

Extraction revues du jeu de données ou publient l'INRAE et le MPI

```
1) val common = artinraeudf.select($"Title").distinct()
                        .join(artmpiudf.select($"Title").distinct(),Seq("Title"),"inner")
common.show()
+------------------+
|             Title|
+------------------+
|Scientific reports|
+------------------+

2) val common = artinraeudf.select($"Title").distinct()
    .join(artmpiudf.select($"Title").distinct(),artinraeudf("Title") ===
    artmpiudf("Title"),"inner")
+------------------+------------------+
|             Title|             Title|
+------------------+------------------+
|Scientific reports|Scientific reports|
+------------------+------------------+

3) val common = artinraeudf.select($"Title").distinct()
    .join(artmpiudf.select($"Title").distinct(),artinraeudf("Title") ===
    artmpiudf("Title"),"inner")
    .drop(artmpiudf("Title")) <- suppression de la colonne dupliquée en spécifiant son origine
```

Note: je n'ai pris qu'un seul xml pour le ppt!

Pivoter sur une ou plusieurs colonnes, appliquer des fonctions d'agrégations (ex: count) et de tri

```
artinraeudf.groupBy($"isFrom",$"Title").pivot("Year").count.sort(desc("2019")).show(20,40)
+------+----------------------------------------+----+----+
|isFrom|                                   Title|2019|2020|
+------+----------------------------------------+----+----+
|  true|                    Scientific reports|   2|null|
|  true|                             Nutrients|   2|null|
|  true|      Journal of clinical epidemiology|   1|null|
|  true|  Genetics, selection, evolution : GSE|   1|null|
|  true|   Transboundary and emerging diseases|   1|null|
|  true|        Journal of theoretical biology|   1|null|
|  true|                          Food chemistry|   1|   1|
|  true|               Frontiers in immunology|   1|null|
|  true|             Environment international|   1|null|
|  true|Journal of colloid and interface science|   1|null|
|  true|International journal of molecular sc...|   1|null|
|  true|The Plant journal : for cell and mole...|   1|null|
|  true|                           Data in brief|   1|null|
|  true|                         Theriogenology|   1|null|
|  true|Applied microbiology and biotechnology|   1|null|
|  true|Journal of physiology and biochemistry|   1|null|
|  true|                                   Cells|   1|null|
|  true|                      Annals of botany|null|   1|
|  true|                Pest management science|null|   1|
|  true|                    The New phytologist|null|   1|
+------+----------------------------------------+----+----+
```

On peut définir ses propres fonctions d'agrégations
User Defined Aggregate Function: UDAF

Exploser les structures complexes (Array, Map, etc.) en lignes

Compter le nombre de publications par auteur

```
val authexpl = artft.select($"Title",$"Year",explode($"Author")) ← une ligne Title,Year pour
                                                                  tous les auteurs de la publi

authexpl.show(5)
+--------------------+----+--------------------+
|               Title|Year|                 col|
+--------------------+----+--------------------+
|              Nature|2019|{null, null, Sall...|
|              Nature|2019|{null, null, Susa...|
|ACS synthetic bio...|2019|{[{Discovery Biol...|
|ACS synthetic bio...|2019|{[{Data Sciences ...|
|ACS synthetic bio...|2019|{[{Discovery Biol...|
+--------------------+----+--------------------+

val authexplsel = authexpl.withColumn("Initials",$"col.Initials")
                          .withColumn("LastName",$"col.LastName")

authexplsel.groupBy($"Initials",$"LastName").count.show(5)
+--------+--------+-----+
|Initials|LastName|count|
+--------+--------+-----+
|       F|  Berger|    3|
|       O|  Balafa|    1|
|       B|Theilman|    1|
|      WS|     Kim|    2|
|      HE|    Choi|    1|
+--------+--------+-----+
```

- Essayer de minimiser les transferts de données (ex join), de maximiser les traitements intra partition

Collecte des résultats en mémoire du driver

```
val titlecount = artinrae.groupBy($"Title").count.sort(desc("count"))

titlecount: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] =
    [Title: string, count: bigint]

val titlecountscala = titlecount.limit(5).map(j =>
    (j.getString(0),j.getLong(1))).collect().toMap

titlecountscala: scala.collection.immutable.Map[String,Long] = Map(Applied
    microbiology and biotechnology -> 1, Nutrients -> 2, International
    journal of molecular sciences -> 1, Scientific reports -> 2, Food
    chemistry -> 2)
```

DataFrame vs DataSet

Les RDD originels restent la couche de plus bas niveau.
Très puissants mais il vaut mieux éviter de les utiliser sauf cas très particulier.

- **DataFrame**
  - Disponible dans tous les API/langages de spark
  - Optimisé: type interne Row/Column=Array[Byte]
  - Inférence automatique du schéma à partir des données possible
    - Pratique durant développement mais il vaut mieux le spécifier pour la mise en production
  - Validation à l'exécution
  - DF=Dataset[Row]
- **DataSet**
  - Typé: schema obligatoire à la création
  - Disponible seulement en scala/java
  - Plus lents que les DF (20% d'après « The Definite Guide »)
  - Validation à l'édition (IDEA) / compilation.
  - DS=DF.as[Type]

2e partie

Les dates/timestamp; import/export; définir/spécifier le schema

On repart du début pour sélectionner le champ ArticleDate

```
import com.databricks.spark.xml._

val xmlfiles = "/home/gouzy/Atelier/pubmed22/pubmed22n1000.xml.gz"
// environnement HDFS mais fichier hors partition HDFS
    // val xmlfiles = "file:///home/gouzy/Atelier/pubmed22/pubmed22n*.xml.gz"

val art = spark.read.format("xml").option("rowTag","Article").option("excludeAttribute",
    true).option("compression","gzip").load(xmlfiles)

val artmin =  art.select("ArticleTitle","Journal.Title","Abstract.AbstractText","ArticleDate")

scala> artmin.show(3)
+--------------------+--------------------+--------------------+------------+
|        ArticleTitle|               Title|        AbstractText| ArticleDate|
+--------------------+--------------------+--------------------+------------+
|Why a right to li...|           Bioethics|[Joona Räsänen ha...|{6, 8, 2019}|
|Cardiac sarcoidos...|Echocardiography ...|[Genetic factors ...|{7, 8, 2019}|
|The Obvious in a ...|Berichte zur Wiss...|[The scope and mi...|{7, 8, 2019}|
+--------------------+--------------------+--------------------+------------+

artmin.select($"ArticleDate").printSchema
root
 |-- ArticleDate: struct (nullable = true)
 |    |-- Day: long (nullable = true)        ← il y a trouvé des valeurs "null"
 |    |-- Month: long (nullable = true)
 |    |-- Year: long (nullable = true)
```

"Cast" des dates en timestamp

```
val fmtDate = "yyyy-M-d"
val artdate = artmin.withColumn("DateRaw",concat($"ArticleDate.Year",lit("-"),$"ArticleDate.Month",lit("-
    "),$"ArticleDate.Day"))
                    .withColumn("Date",to_timestamp($"DateRaw",fmtDate))
artdate.show(3)
+--------------------+--------------------+--------------------+-----------+--------+-------------------+
|        ArticleTitle|               Title|        AbstractText| ArticleDate| DateRaw|               Date|
+--------------------+--------------------+--------------------+-----------+--------+-------------------+
|Why a right to li...|           Bioethics|[Joona Räsänen ha...|{6, 8, 2019}|2019-8-6|2019-08-06 00:00:00|
|Cardiac sarcoidos...|Echocardiography ...|[Genetic factors ...|{7, 8, 2019}|2019-8-7|2019-08-07 00:00:00|
|The Obvious in a ...|Berichte zur Wiss...|[The scope and mi...|{7, 8, 2019}|2019-8-7|2019-08-07 00:00:00|
+--------------------+--------------------+--------------------+-----------+--------+-------------------+
val artdatemin = artdate.withColumn("Abstract",lower($"AbstractText"(0)))
    .withColumn("Journal",lower($"Title"))
    .drop($"AbstractText").drop($"ArticleDate").drop($"DateRaw").drop($"Title").drop($"ArticleTitle")
    // .select($"Date",$"Abstract",$"Journal")
    .na.drop()
artdatemin.show(3)
+-------------------+--------------------+--------------------+
|               Date|            Abstract|             Journal|
+-------------------+--------------------+--------------------+
|2019-08-06 00:00:00|joona räsänen has...|           bioethics|
|2019-08-07 00:00:00|genetic factors p...|echocardiography ...|
|2019-08-07 00:00:00|the scope and mis...|berichte zur wiss...|
+-------------------+--------------------+--------------------+
artdatemin.printSchema()
root
 |-- Date: timestamp (nullable = true)
 |-- Abstract: string (nullable = true)
 |-- Journal: string (nullable = true)
```

Export au format parquet dans un dossier à date

```
val today = spark.range(1).select(current_date).head.get(0).toString  ← date du jour dans une String
today: String = 2022-11-21

val prefix = "ARTICLE_MIN_"

artdatemin.write.format("parquet").mode("overwrite").save(prefix+today) ← par défaut n'écrase pas

import scala.sys.process._

val cmdfind = "find "+prefix+today+" -ls"
cmdfind: String = find ARTICLE_MIN_2022-11-21 -ls

cmdfind.!!.split("\n").foreach(println)   ← execution de la commande système hors HDFS
 28443988      4 drwxr-xr-x   2 gouzy    gouzy        4096 nov. 21 11:03 ARTICLE_MIN_2022-11-21
 28714650     60 -rw-r--r--   1 gouzy    gouzy       61400 nov. 21 11:03 ARTICLE_MIN_2022-11-21/.part-00000-
    82f034ac-c546-44b6-ad8d-8d3e930d56c0-c000.snappy.parquet.crc
 28714647   7676 -rw-r--r--   1 gouzy    gouzy     7858150 nov. 21 11:03 ARTICLE_MIN_2022-11-21/part-00000-
    82f034ac-c546-44b6-ad8d-8d3e930d56c0-c000.snappy.parquet
 28443989      0 -rw-r--r--   1 gouzy    gouzy           0 nov. 21 11:03 ARTICLE_MIN_2022-11-21/_SUCCESS
 28443990      4 -rw-r--r--   1 gouzy
```

Format Parquet

Stockage ligne (ex: csv) vs colonne (ex: parquet): filtre plus simple, compression plus efficace

Table données

	Col1	Col2	Col3	Col4
Ligne1	L1C1	L1C2	L1C3	L1C4
Ligne2	L2C1	L2C2	L2C3	L2C4
Ligne3	L3C1	L3C2	L3C3	L3C4

Stockage par ligne

L1C1	L1C2	L1C3	L1C4	L2C1	L2C2
L2C3	L2C4	L3C1	L3C2	L3C3	L3C4

Stockage par colonne

L1C1	L2C1	L3C1	L1C2	L2C2	L3C2
L1C3	L2C3	L3C3	L1C4	L2C4	L3C4

➔ plus optimisé (moins d'accès disques) pour des requêtes du type .select($"Col2").where($"Col2" > X)

Lecture données en spécifiant un schéma

Contrôle intégrité & Changement de nom

```
import org.apache.spark.sql.types.{StringType, StructField, StructType,
    TimestampType}

val schema = new StructType()
    .add(StructField("date",TimestampType,true))    ← on change le nom
     .add(StructField("abstract",StringType,true))
     .add(StructField("journal",StringType,true))

val restart = spark.read.schema(schema).load(prefix+today)   <- défaut parquet

scala> restart.show(3)
+-------------------+--------------------+--------------------+
|               date|            abstract|             journal|
+-------------------+--------------------+--------------------+
|2019-08-06 00:00:00|joona räsänen has...|           bioethics|
|2019-08-07 00:00:00|genetic factors p...|echocardiography ...|
|2019-08-07 00:00:00|the scope and mis...|berichte zur wiss...|
+-------------------+--------------------+--------------------+
```

Annotation & analyse temporelle

Analyse basique de la dynamique de publication "plant" et "animal"

```
val artannot = restart.withColumn("isPlant",lower($"abstract".contains(" plant "))
                      .withColumn("isAnimal",$"abstract".contains(" animal "))

artannot.show(8)
+-------------------+--------------------+--------------------+-------+--------+
|               date|            abstract|             journal|isPlant|isAnimal|
+-------------------+--------------------+--------------------+-------+--------+
|2019-08-06 00:00:00|joona räsänen has...|           bioethics|  false|   false|
|2019-08-07 00:00:00|genetic factors p...|echocardiography ...|  false|   false|
|2019-08-07 00:00:00|the scope and mis...|berichte zur wiss...|  false|   false|
|2019-08-06 00:00:00|inadequate sleep ...|the journal of ph...|  false|   false|
|2019-08-06 00:00:00|smokers are less ...|alimentary pharma...|  false|   false|
|2019-08-06 00:00:00|a 75-year-old man...|the journal of de...|  false|   false|
|2019-08-07 00:00:00|effect of fonio-m...|journal of food b...|  false|   false|
|2019-08-06 00:00:00|montelukast (mnk)...|drug development ...|  false|    true|
+-------------------+--------------------+--------------------+-------+--------+
val dyna = artannot.groupBy(window($"date","100 days"),$"isPlant",$"isAnimal")
                   .count.sort(desc("window"))
dyna.show(8,50)
+------------------------------------------+-------+--------+-----+
|                                    window|isPlant|isAnimal|count|
+------------------------------------------+-------+--------+-----+
|{2021-03-14 01:00:00, 2021-06-22 02:00:00}|  false|   false|    6|
|{2020-12-04 01:00:00, 2021-03-14 01:00:00}|  false|   false|    7|
|{2020-08-26 02:00:00, 2020-12-04 01:00:00}|  false|   false|    2|
|{2020-05-18 02:00:00, 2020-08-26 02:00:00}|   true|   false|    1|
|{2020-05-18 02:00:00, 2020-08-26 02:00:00}|  false|   false|    2|
|{2020-02-08 01:00:00, 2020-05-18 02:00:00}|  false|   false|    2|
|{2019-10-31 01:00:00, 2020-02-08 01:00:00}|   true|   false|    3|
|{2019-10-31 01:00:00, 2020-02-08 01:00:00}|  false|    true|    1|
+------------------------------------------+-------+--------+-----+
```

Export CSV avec partitionnement

1) pas de structure de données complexes (Struct/Array/Map) en CSV
2) df.coalesce(1).write... permet de ne créer qu'un seul fichier csv (sans partionnement)

```
val outcsvdir ="ARTICLE_ANNOT_CSV"+today
artannot.write.format("csv").partitionBy("isPlant","isAnimal").mode("overwrite").save(outcsvdir)

val cmdfind = "find "+outcsvdir+" -ls"
cmdfind.!!.split("\n").foreach(println)
ARTICLE_ANNOT_CSV2022-11-21
ARTICLE_ANNOT_CSV2022-11-21/isPlant=true
ARTICLE_ANNOT_CSV2022-11-21/isPlant=true/isAnimal=false
ARTICLE_ANNOT_CSV2022-11-21/isPlant=true/isAnimal=false/part-00000-0c27ed3b-6625-4bef-a10d-3f267a61c5ab.c000.csv
ARTICLE_ANNOT_CSV2022-11-21/isPlant=true/isAnimal=false/.part-00000-0c27ed3b-6625-4bef-a10d-
    3f267a61c5ab.c000.csv.crc
ARTICLE_ANNOT_CSV2022-11-21/isPlant=true/isAnimal=true
ARTICLE_ANNOT_CSV2022-11-21/isPlant=true/isAnimal=true/part-00000-0c27ed3b-6625-4bef-a10d-3f267a61c5ab.c000.csv
ARTICLE_ANNOT_CSV2022-11-21/isPlant=true/isAnimal=true/.part-00000-0c27ed3b-6625-4bef-a10d-
    3f267a61c5ab.c000.csv.crc
ARTICLE_ANNOT_CSV2022-11-21/_SUCCESS
ARTICLE_ANNOT_CSV2022-11-21/._SUCCESS.crc
ARTICLE_ANNOT_CSV2022-11-21/isPlant=false
ARTICLE_ANNOT_CSV2022-11-21/isPlant=false/isAnimal=false
ARTICLE_ANNOT_CSV2022-11-21/isPlant=false/isAnimal=false/part-00000-0c27ed3b-6625-4bef-a10d-3f267a61c5ab.c000.csv
ARTICLE_ANNOT_CSV2022-11-21/isPlant=false/isAnimal=false/.part-00000-0c27ed3b-6625-4bef-a10d-
    3f267a61c5ab.c000.csv.crc
ARTICLE_ANNOT_CSV2022-11-21/isPlant=false/isAnimal=true
ARTICLE_ANNOT_CSV2022-11-21/isPlant=false/isAnimal=true/part-00000-0c27ed3b-6625-4bef-a10d-3f267a61c5ab.c000.csv
ARTICLE_ANNOT_CSV2022-11-21/isPlant=false/isAnimal=true/.part-00000-0c27ed3b-6625-4bef-a10d-
    3f267a61c5ab.c000.csv.crc
```

Ressources importantes et utiles

à regarder absolument

Spark built in functions

- https://spark.apache.org/docs/3.3.1/api/scala/org/apache/spark/sql/functions$.html

Import direct de Feuilles Excel

```
spark-shell --packages com.crealytics:spark-excel_2.12:3.3.1_0.18.5
val f1=spark.read.format("excel").option("header",true)
             .option("dataAddress","1!A1") <- import feuille 2
             .load("/home/gouzy/LiuCell2020-PRJCA002030.CRA002269.xlsx")
```

Connecteurs bases de données

Travaux Pratiques

Durée: 2h

- **Codez et exécutez dans le spark-shell les 2 exemples présentés précédemment**
  - **n'hésitez pas à tester l'auto-complétion des méthodes (<TAB>) disponibles à partir de vos structures de données**
  - **Attention:**
    - **utilisez le nom du journal pour la classification Plant/Animal (pas l'abstract comme dans le cours)**
    - **générer la classification "isPlant" sera utilisé pour l'atelier ML**
    - **sauvegardez un dataframe seulement avec "abstract" et "isPlant"**
- **Codez les analyses suivantes**
  - **Combien de publications sont cosignées par l'INRAE et le Max Planck ?**
  - **Quelle est la période (100j) ou l'INRAE et le Max Planck ont le plus de publications en commun ?**
  - **Quels sont les mots de plus de 5 caractères les plus fréquents dans les titres des publications INRAE**
- **Pour répondre à ces questions**
  - **développez sur votre ordinateur en utilisant 2 ou 3 fichiers XML (avec "*").**
    ex:
    - `scp login@138.102.223.150:/mnt/shared/data/pubmed22/pubmed22n1000.xml.gz .`
  - **une fois que votre programme sera fonctionnel, vous le testez grandeur réelle (en supprimant les .show()), en chargeant le dossier parquet, en écrivant les résultats dans un dossier en csv**

Complément d'informations pour le TP

- fichier pubmed22n1000.xml.gz également dispo dans nextcloud/TP
- pubmed22 complet est disponible sur le cluster idf – 138.102.223.150
  - /mnt/shared/data/pubmed22
- pour répondre aussi bien au cours qu'aux questions du TP en pg 8:

```
val artmin =

art.select("ArticleTitle","Journal.Title","Journal.JournalIssue.PubDate.Year","AuthorList.Author","Abstract.AbstractText","ArticleDate"))
```

- val artmin = ...persist() dès l'initialisation si la lecture du fichier xml est longue
- sur le cluster idf, écrivez vos résultats dans /mnt/shared/data/$USER
- exécution sur cluster idf
  - `spark-shell --packages com.databricks:spark-xml_2.12:0.15.0 --master "spark://10.0.0.184:7077" --total-executor-cores 8`

- `TP ML (gros) : 138.102.223.150:/mnt/shared/data/PUBMED4ML`

Code pour générer /mnt/shared/data/PUBMED4ATELIER

25mn sur 16 coeurs (cluster idf) pour transformer 35Gb xml (compressés) en 533Mb parquet

```
import com.databricks.spark.xml._
import org.apache.spark.sql.{Column, Row}
import org.apache.spark.sql.types.{StringType, StructField, StructType, TimestampType}
import scala.util.matching.Regex
val xmlfiles = "file:///mnt/shared/data/pubmed22/pubmed22n*.xml.gz"

val art = spark.read.format("xml").option("rowTag","Article").option("excludeAttribute", true)
                                  .option("samplingRatio",0.1).option("compression","gzip").load(xmlfiles)
val artmin = art.select("ArticleTitle","Journal.Title","Abstract.AbstractText","ArticleDate","AuthorList.Author",
     "Journal.JournalIssue.PubDate.Year").na.drop()

def isStringOK(str: String,pattern:String): Boolean = { val regex: Regex = pattern.toLowerCase().r; if
     (regex.findFirstIn(str.toLowerCase()).size > 0) { return true }; return false }
val isStringOKUDF = udf[Boolean,String,String](isStringOK)

val keywords = "(plant|animal|agriculture|environment)"
val artdomain = artmin.withColumn("DomainPattern",lit(keywords))
                      .withColumn("isDomainOK",isStringOKUDF($"Title",$"DomainPattern"))
val instituts = "(INRA|IRSTEA|CEMAGREF|MPI|Max Planck Institute|Wageningen University|USDA)"
def isOriginOK(authors: Seq[Row],instituts:String): Boolean = { val regex: Regex = instituts.toLowerCase().r; val
     l_affiliations = authors.map(auth => auth.getSeq(0)).toList.filter(aff => aff != null); if (
     l_affiliations.isEmpty ) { return false}; val s_affiliations = l_affiliations.flatten.toString().toLowerCase();
     if (regex.findFirstIn(s_affiliations).size > 0 ) { return true }; return false }
val isOriginOKUDF = udf[Boolean,Seq[Row],String](isOriginOK)
val artinstituts = artdomain.withColumn("OriginPattern",lit(instituts))
                            .withColumn("isOriginOK",isOriginOKUDF($"Author",$"OriginPattern"))

val artselect= artinstituts.where("isOriginOK=true or isDomainOK=true" )
                           .drop($"isDomainOK").drop($"isOriginOK").drop($"OriginPattern").drop($"DomainPattern")
artselect.write.format("parquet").mode("overwrite").save("/mnt/shared/data/PUBMED4ATELIER")
```