Big data batch processing
M2 LID and SIA
2022-23

Triples manipulation with Spark Core and in the Databricks Community edition

1 Triples manipulation

In this question, we are going to manipulate a set of tuples of size 3. We will call them triples and they enable us to represent a graph. In fact, the elements of the triple are respectively named a subject, a predicate and an object. Hence, the triple (1,2,3) states that a node graph identified by id=1 is related to node graph with id=3 via the predicate identified with the value 2.

Our running example corresponds to the following set of tuples:

((1,0,5),(5,1,8),(8,2,1),(2,0,6),(3,0,6),(6,1,9),(5,1,9),(9,3,11),(9,4,12),(4,0,7),(7,1,9),(7,2,10),(14,1,15), (15,1,16),(14,1,16),(17,0,18),(18,0,19),(19,1,20),(20,0,17))

Draw this graph in order to easily verify your answers to the following questions.

1.1

Create a new Scala object (in the same project as Question 1) that creates an RDD, named triples, from our running example set of triples.

1.2

From the RDD in the previous section, create an RDD, named soPairs, that contains only the subject and objects of the triples.

1.3

Using soPairs, find the graph nodes which correspond to roots of the graph (no in-going edges on these graph nodes). Name that RDD as roots.

1.4

Create an RDD, denoted leaves, that contains the ids of the graph nodes corresponding to leaves (no out-going edges).

1.5

Create a new RDD that contains the set of nodes accessible from each node of the graph. This corresponds to the transitive closure of the graph). Example, given a $\rightarrow$ b $\rightarrow$ c, we will end up with (a,b),(b,c), which we already had, and (a,c). *Help: You can use the join RDD operation to perform this task.*

1.6

Display, in sorted order on the subject value, only the subject,object pairs that have been added in the transitive closure (that is the pairs that were not originally in soPairs).

1.7

Given the RDDs created so far, create an RDD, named rooted, which contains the set of nodes accessible from a root. The resulting RDD should look like a tuple with the root in the first position and a sorted list of all accessible nodes in the second position.

1.8

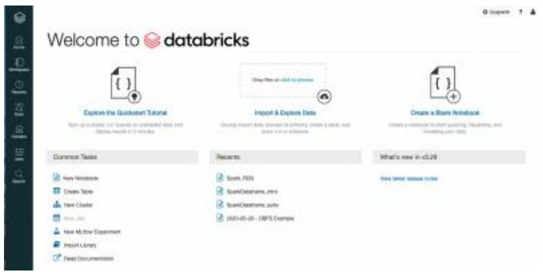
Create an RDD, denoted cycles, that contains this graph's set of cycles (that is its set of graph nodes).

2 Introduction to Databricks Community Edition

The Databricks Community Edition (DCE) offers to use a Spark notebook on its Cloud. You first need to create an account at https://community.cloud.databricks.com/login.html.

DCE provides storage facilities for your notebooks as well as a place to store your datasets (dbfs).

Once you have created an account and you are connected to it, the main page looks as follows:



The sidebar offers several features:

- "Worskpace" to manage your notebooks and configuration files
- "Data" to import some datasets
- "Clusters" to create, start and stop a Clusters
- "Jobs" is reserved for paying accounts and permits to visualize metrics during the execution of programs

2.1 Create a cluster

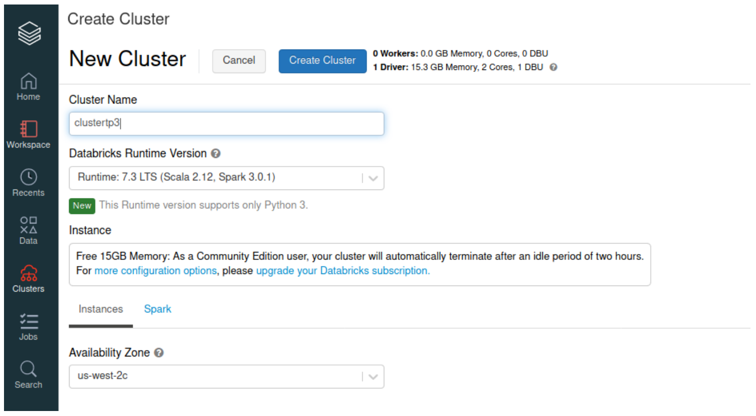
For you first execution, click on create a cluster, provide a name to it and keep the default parameters. After few seconds (or minutes), the cluster is ready. It will be killed after 2 hours of inactivity. The cluster corresponds to a machine with 15GB of RAM, 2 cores and 1 database unit. Removing the cluster does not remove your data, nor your notebook which are stored in the Databricks storage space.

2.2 Data loading

You can load some data by clicking on the "data" tab and then on "Add data" on the sidebar. Create a folder to organize your data by typing a relative path in the text area.

Effectively load the data via drag and drop in the Files zone. Then, the full path should appear. It is the path that needs to be specified in the scripts you will write in the Notebook. You can remove it otherwise, it will be stored in the DBFS.

For instance, load the mobyDick.txt file. Copy the complete file path.



Create Cluster

New Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU
1 Driver: 15.3 GB Memory, 2 Cores, 1 DBU

Cluster Name: clustertp3

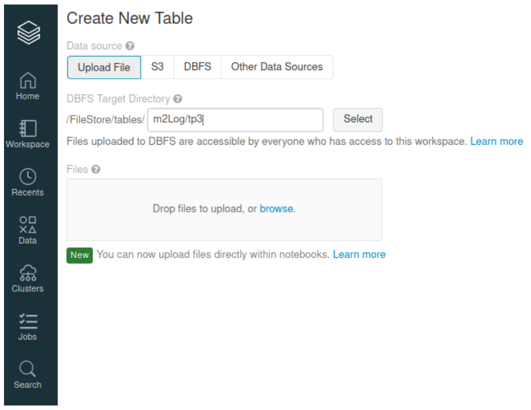
Databricks Runtime Version: Runtime: 7.3 LTS (Scala 2.12, Spark 3.0.1)

New This Runtime version supports only Python 3.

Instance

Free 15GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours.
For more configuration options, please upgrade your Databricks subscription.

Availability Zone: us-west-2c



Create New Table

Data source: Upload File

DBFS Target Directory: /FileStore/tables/ m2Log/tp3

Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more

Files

Drop files to upload, or browse.

New You can now upload files directly within notebooks. Learn more

2.3 Create a notebook

In the sidebar's menu, select Workspace to get a new menu offer. Click on the arrow of your name. You can then create, import or clone a notebook.

Fill the form by typing a notebook name and selecting a programing language (Python is the default option, you should select Scala).

Once in the notebook, type the following Scala code:

```
val a = List(1,2,3,4,5)
val b = sc.parallelize(a).map(x=>x*x)
b.take(5)
```

An run it by using the 'play' symbol on the right handside of Cmd1.

Create another Cmd box by clicking on (+) below the current Cmd box. and test the following Scala code: Note that this makes reference to the data I have previously created. You should modify the path of the file.

```
 val moby = sc.textFile("/FileStore/tables/m2Log/tp3/mobyDick.txt")
moby.count
```

3 Materialization of sameAs properties

As in the previous lab session, we are still considering triples of the form (subject, predicate, object). This time they are represented as strings. In our running example, one of the predicates is "sameAs". It states that the subject and object related by this predicate are corresponding to same entity. The property is symmetric and transitive. That is, if we consider the following example:

```
a sameAs b
b sameAs c
```

We can deduce the following four additional triples:

```
1. b sameAs a
2. c sameAs b
3. a sameAs c
4. c sameAs a
```

The deductions 1 and 2 are following from symmetry, 3 from transitivy and 4 from symmetry using the triple 3.

In a system that handles such triples, one way to handle those deductions is to materialize them in the database or the orginal file.

Usually, such database systems encode the entries of the triples with integer values because it reduces the memory footprint and matching operations are efficiently performed.

You are given a data set (drugBankExt.nt) and a set of dictionaries for property and individual entries.

3.1

Encode the drugBankExt.nt triples with the dictionaries. That is, we will end up with triples composed of 3 integer values.

3.2

Materialize all deductions that can be performed on the sameAs triples.