Mini guida - Data Lake Brescia Musei

Installazione:

Repository: [stefanopoma-97/data_lake_brescia_musei (github.com)]

Per l'installazione seguire passo passo tutti i punti riportati di seguito (presenti anche nel README della repository). Il progetto è stato testato in ambiente Windows e Mac.

JAVA

1. Scaricare Java 11: https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html
2. Installare e inserire in: C:\Program Files\Java\jdk-11.0.15.1\bin
3. Configurare variabile di sistema JAVA_HOME : C:\Program Files\Java\jdk-11.0.15.1
4. Configurare variabile PATH : %JAVA_HOME%\bin (tutorial: https://www.ibm.com/docs/en/b2b-integrator/5.2?topic=installation-setting-java-variables-in-windows)
5. In C:\Program Files\Common Files\Oracle\Java Rimuovere il collegamento javapath

PYCHARM

1. Installare PyCharm: https://www.jetbrains.com/help/pycharm/installation-guide.html
2. Configrare interprete Python 3.9 (tutorial: https://www.jetbrains.com/help/pycharm/creating-virtual-environment.html)

HADOOP

1. Creare C:\hadoop
2. Su questo sito: https://archive.apache.org/dist/hadoop/common/ Scaricare una versione di Hadoop (testato con la 3.3.1)
3. Scaricare 2 file: hadoop-3.3.1-src.tar.gz e hadoop-3.3.1.tar.gz.
4. Inserire i file nella cartella create nel seguente modo: C:\hadoop\hadoop-3.3.1\bin C:\hadoop\hadoop-3.3.1-src
5. Scaricare winutils.exe dal seguente link https://github.com/kontext-tech/winutils/tree/master/hadoop-3.1.1/bin

6. Inserire il file scaricato in C:\hadoop\hadoop-3.3.1\bin\
7. Scaricare dallo stesso link il file hadoop.dll e inserirlo in C:\Windows\System32

SPARK

1. Scaricare Spark da https://spark.apache.org/downloads.html (testato con Spark 3.3.)
2. Estrarre il contenuto dell'archivio e inseriere la cartella spark-3.3.0-bin-hadoop3 in C:\
3. Impostare la variabile di sistema SPARK_HOME : C:\spark-3.3.0-bin-hadoop3
4. Impostare la variabile PATH : %SPARK_HOME%\bin
5. Impostare la variabile di sistema HADOOP_HOME : C:\hadoop\hadoop-3.3.1
6. Per testare che l'installazione sia andata a buon fine aprire il terminale e digitare: cd %SPARK_HOME%/bin Successivamente usare il comando "spark-shell"

IMPOSTARE PYSPARK

1. Installare pyspark. Su pycharm andare in: File-> Setting-> Project Name -> Python Interpreter, cliccare sul +, cercare il pacchetto "pyspark" e installarlo
2. Per testare il funzionamento eseguire il seguente script:
   from datetime import datetime, date from pyspark.sql import Row from pyspark import SparkContext from pyspark.sql import SparkSession
   sc = SparkContext("local", "My App") spark = SparkSession.builder.getOrCreate() df = spark.createDataFrame([Row(a=1, b=4., c='GFG1', d=date(2000, 8, 1), e=datetime(2000, 8, 1, 12, 0))]) df.show()

NEO4J

1. File-> Setting-> Project Name -> Python Interpreter, cliccare sul + cercare e installare "neo4j" e "py2neo"
2. nella seguente pagina trovare la versione del connettore in base alla versione di Spark e di Python installata. testato con Scala 2.12 e Spark 3.0+ quindi con il connettore: neo4j-connector-apache-spark_2.12-4.0.1_for_spark_3.jar
3. Sulla seguente pagina scaricare il file jar del connettore individuato: https://github.com/neo4j-contrib/neo4j-spark-connector/releases
4. Inserire il file jar in: C:\spark-3.3.0-bin-hadoop3\spark-3.3.0-bin-hadoop3\bin\neo4j-connector-apache-spark_2.12-4.1.4_for_spark_3.jar
5. Scaricare e installare Neo4j Desktop dal sito ufficiale
6. Eseguire l'applicazione e creare un nuovo progetto chiamato "CMS Brescia Musei"
7. Premere su "Add" e aggiungere un "Local DBMS"
8. Nome: "cms_brescia", Password: "neo4j_cms_brescia"

9. Confermare e attendere che il database venga creato
10. Premere su "Start"
11. Successivamente premere su "Open" per aprire da vista da Browser 1

Commit per testare il funzionamento

Nella repository è presente una commit chiamata “Data Lake PULITO”.

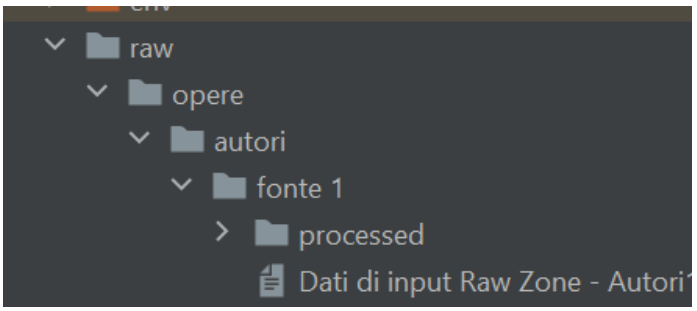
In questa commit i vari file presenti nelle diverse cartelle del data lake sono stati eliminati, mantenendo solamente alcuni file con funzioni di test all’interno della raw zone. Per testare il software partendo da una situazione “pulita” (senza nessuna elaborazione già effettuata) si consiglia quindi di utilizzare questa commit.

Struttura a cartelle

Il data lake è costituito da 3 cartelle principali, rappresentanti le prime 3 zone del data lake:

- raw
- standardized
- curated

Ogni cartella è ulteriormente divisa in due sottocartelle: “opere” e “visitatori”. Dentro le due sottocartelle si trovano ulteriori cartelle rappresentanti i vari tipi di informazioni che possono essere gestite dal software: visite, categorie, autori ecc. La struttura si ramifica ulteriormente introducendo delle ulteriori sottocartelle rappresentanti possibili diverse sorgenti (o fonti).



1 livello: Zone
2 livello: macro divisione Opere e Visitatori (entità core)
3 livello: entità e relazioni secondarie
4 livello: sorgenti/fonti

L’utente non deve modificare tale struttura affinché il software possa funzionare correttamente.
Per inserire dei nuovi file all’interno della raw zone si deve semplicemente aggiungere una cartella di livello 4 (una sorgente) ed inserire i file all’interno di quest’ultima (è anche possibile inserire file all’interno di una cartella di livello 4 già presente).

Testare gli script

Gli script per gestire l'evoluzione dei dati all'interno del data lake sono 3:

- raw_zone.py: fa evolvere i file dalla raw zone alla standardized zone.
- standardized_zone.py: fa evolvere i file dalla standardized zone alla curated zone.
- curated_zone.py: fa evolvere i file dalla curated zone all'application zone (db a grafo).

Per verificare quindi la corretta evoluzione del dato i 3 script dovrebbero essere eseguiti in sequenza.

Ogni script permette anche di far evolvere solamente certi file e non per forza l'intera zona. Questo viene realizzato tramite un menù con una serie di opzioni:

```
Standardized -> Curated
Seleziona un'opzione:
1) Opere
2) Descrizioni
3) Autori
4) Immagini
5) Categorie
6) Visitatori
7) Visite
0) Tutti
```

Si consiglia di selezionare un'opzione da 1 a 7 qualora si stia testando il funzionamento di una particolare funzione. L'opzione 0 serve invece a far evolvere i dati dell'intera zona.

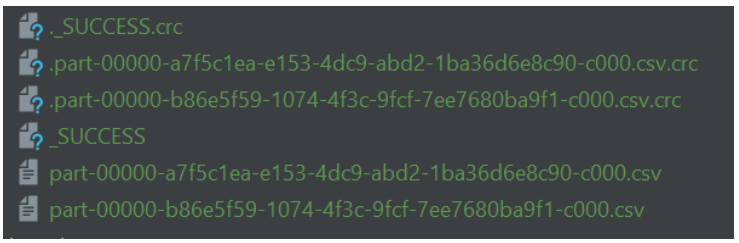
Il menù dello script curated_zone.py è leggermente diverso:

```
Standardized -> Curated
Seleziona un'opzione:
1) Categoria e visitatore
2) Opera, autore, descirzione e immagini
3) Visite, visitatore e opera
4) Tutti
5) Drop DB
```

Le prime 3 opzioni non vanno a salvare alcuna informazione all'interno del grafo, ma servono solamente a mostrare a video le elaborazioni effettuate.
L'opzione 4 compie tutte le elaborazioni e poi scrive i dati nel grafo.
L'opzione 5 serve a cancellare tutti i nodi e le relazioni dal grafo.

Modifica o aggiunta manuale di file

Per testare il funzionamento di tutti gli script potrebbe essere utile modificare o aggiungere dei file anche nella standardized zone o nella curated zone.
A tal riguardo si consiglia di entrare nella cartella della sorgente dove si vuole aggiungere/modificare un file ed eliminare preventivamente tutti i file non .csv generati automaticamente da Spark.
Nell'esempio sotto riportato, per esempio, bisognerà rimuovere entrambi i file .crc e i file _SUCCESS. Una volta fatto sarà possibile modificare i file .csv o aggiungere altri file senza che Spark vada in errore.



Comunicazione con Neo4J

Affinchè l'ultimo script funzioni è essenziale che sia mantenuta aperta un'istanza di Neo4j. Le configurazioni (nome database e password) sono riportate sopra.