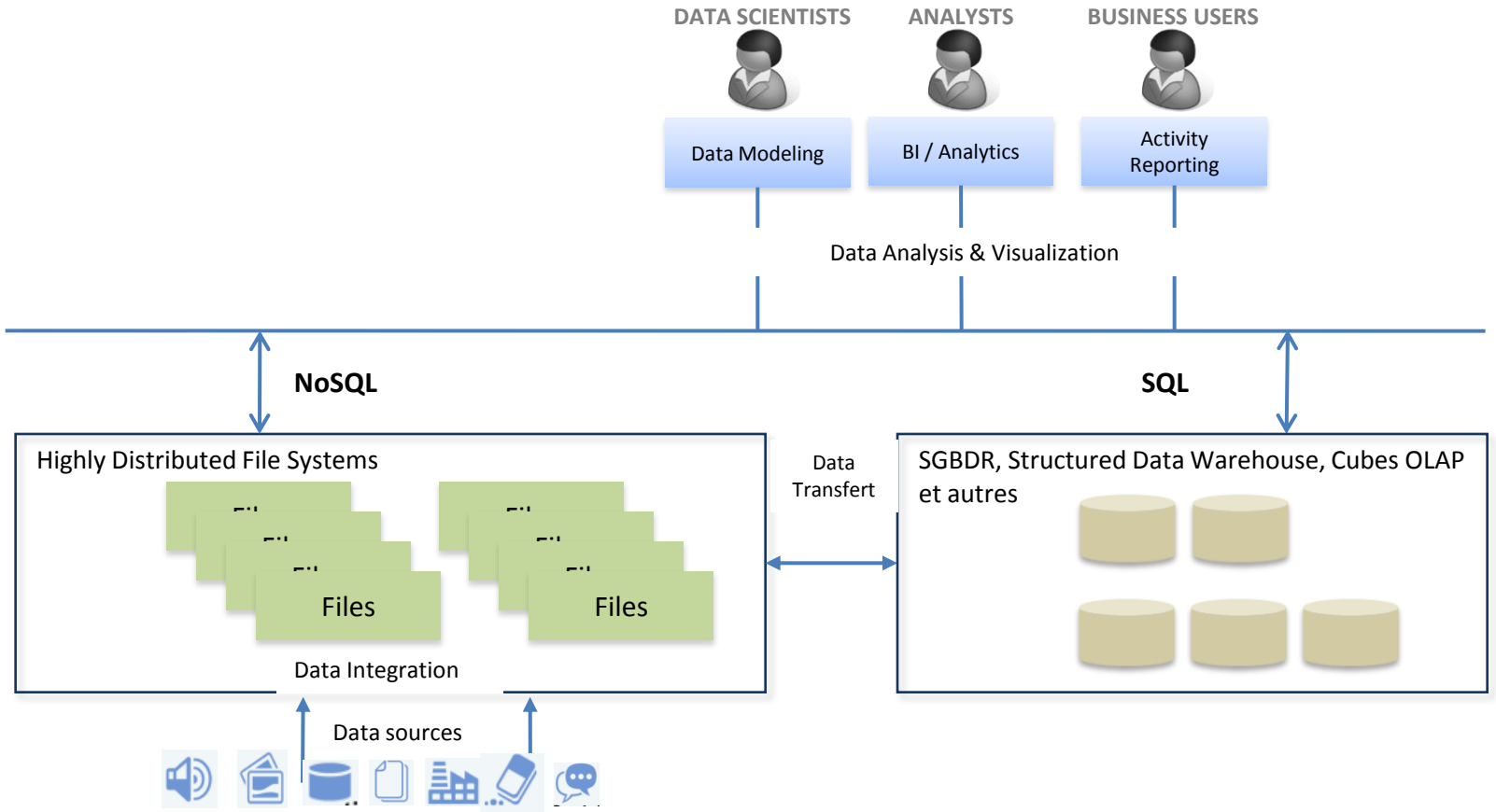
Comment marche le big data???

A part être un sujet marketing faisant couler des flots d'encre digitale, le big data, ce sont des concepts, des techniques.
Le jour est venu pour appréhender en profondeur cet ensemble.
Tout d'abord, voici un schéma du fonctionnement global de la machine:



Le schéma précédent regroupe les hébergements de données en deux groupes principaux: les **données structurées** et les **données non structurées**. Volontairement, et parce que nous connaissons le domaine, nous n'allons pas nous attarder sur les données structurées. Elles sont généralement stockées sur des appliances ou autres SGBDR, alimentées par des ETL et consultées via requêtes SQL.

Les données non structurées sont stockées sur des disques de type HDFS (**Hadoop Distributed File System**). Ces disques permettent une disponibilité améliorée et des performances très en avance par rapport aux disques traditionels.
Sur ceux-ci, les données sont stockées sous forme de fichiers.

L'interrogation de ces données se fait via un ensemble de classes **JAVA**, nommé **Map Reduce.** Nous avons affaire à deux fonctions principales, qui vont assurer la répartition des travaux (Map) et la collecte des résultats (Reduce).
L'implémentation de 'requetes' dans cet environnement peut donner des codes comme dans l'exemple ci-dessous:

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class WordCount {

  public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text,
IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
```

```
    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        output.collect(word, one);
      }
    }
  }

  public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text,
IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) {
        sum += values.next().get();
      }
      output.collect(key, new IntWritable(sum));
    }
  }
```

```
  public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("wordcount");

    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);

    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);

    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);

    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
  }
}
```

L'utilisation de classes JAVA m'étant tout à fait étrangère, je me suis posé la question suivante : Y a-t-il une surcouche plus compréhensible??
Eh bien, vu je ne suis pas le seul à poser cette question, la réponse ne pouvait pas être non. L'interrogation de données est vulgarisée!! Cool…

Y a-t-il une surcouche compréhensible?
Les surcouches natives de Map Reduce sont nommées **PIG** et **HIVE**. Elles permettent de rendre plus compréhensible l'interrogation des données.

Hive est un logiciel d'analyse de données permettant d'utiliser Hadoop avec une syntaxe proche du SQL. Hive a été initialement développé par Facebook.
Pig est un logiciel d'analyse de données comparable à Hive, mais qui utilise le langage Pig Latin. Pig a été initialement développé par Yahoo!

Portant mon dévolu sur le langage le plus proche du mien, voici, ci-dessous une exemple de Hive.
Ce langage est ensuite compilé et transformé en un ensemble de commandes map reduce correspondantes.

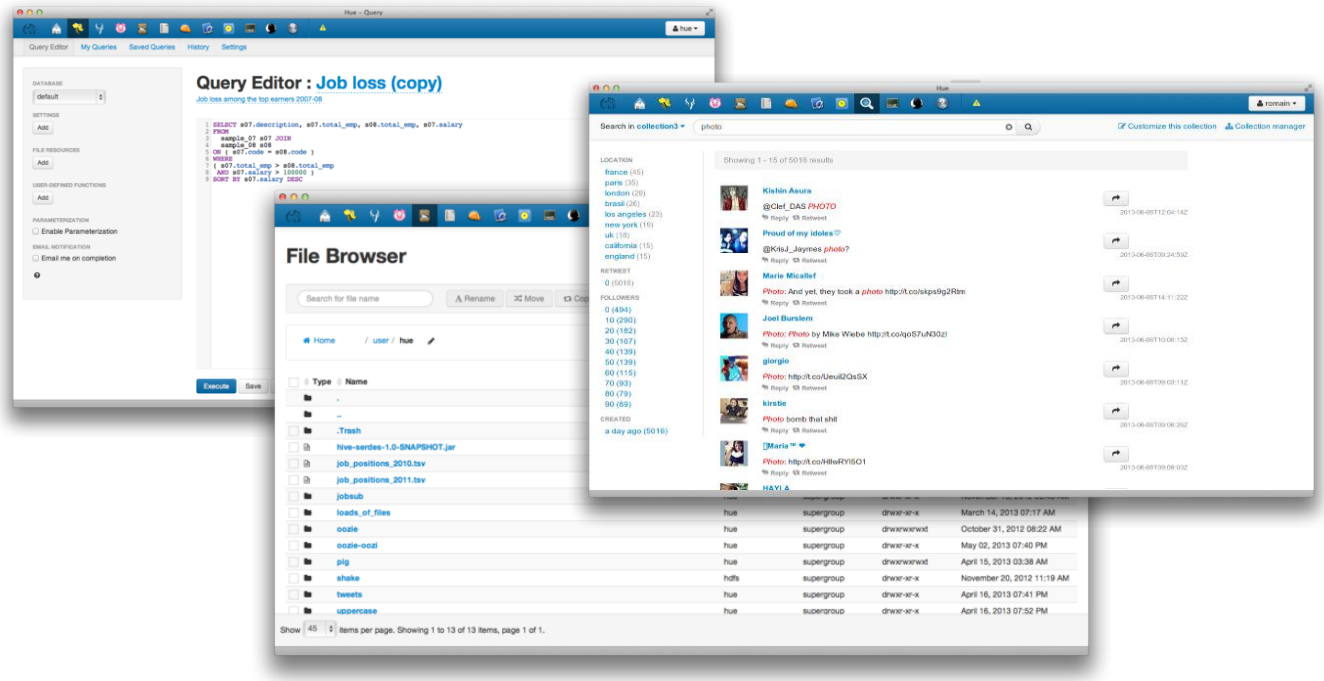
```
CREATE TABLE u_data ( userid INT, movieid INT, rating INT, unixtime STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH 'ml-data/u.data'
OVERWRITE INTO TABLE u_data;

SELECT COUNT(*) FROM u_data;
```

Je me sens plus à l'aise avec ce langage, mais on peut remarquer que l'utilisation exclusive de chargement de tables sera restrictive. Pour cela, l'utilisation conjointe de HIVE et Map Reduce permet de jouer sur les deux tableaux.
Par ailleurs, des collections de classes JAVA sont disponibles permettant de faire toutes sortes d'opérations sur les fichiers.

Y a-t-il un interface user friendly?

Nous avons vu les langages d'interrogation de données, ils peuvent être utilisés via je JDK ou en ligne de commande, rien de bien beau. Le rêve serait d'avoir un outil plus graphique pour éditer les requêtes.

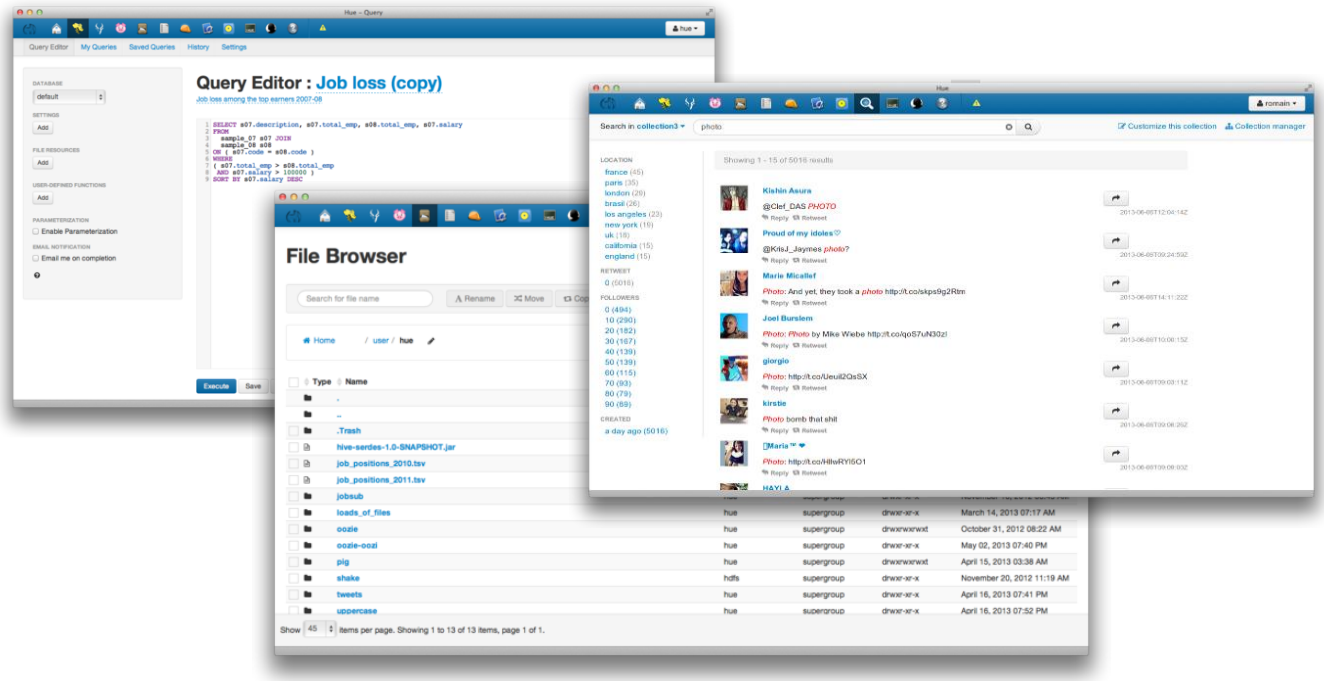
Un outil open source peut s'occuper de ceci, et apporter un environnement graphique pour gérer ses fichiers, éditer et gérer ses requêtes et tout et tout. Cet outil se nomme **HUE**, voici un petit aperçu:



Y a-t-il un interface user friendly?

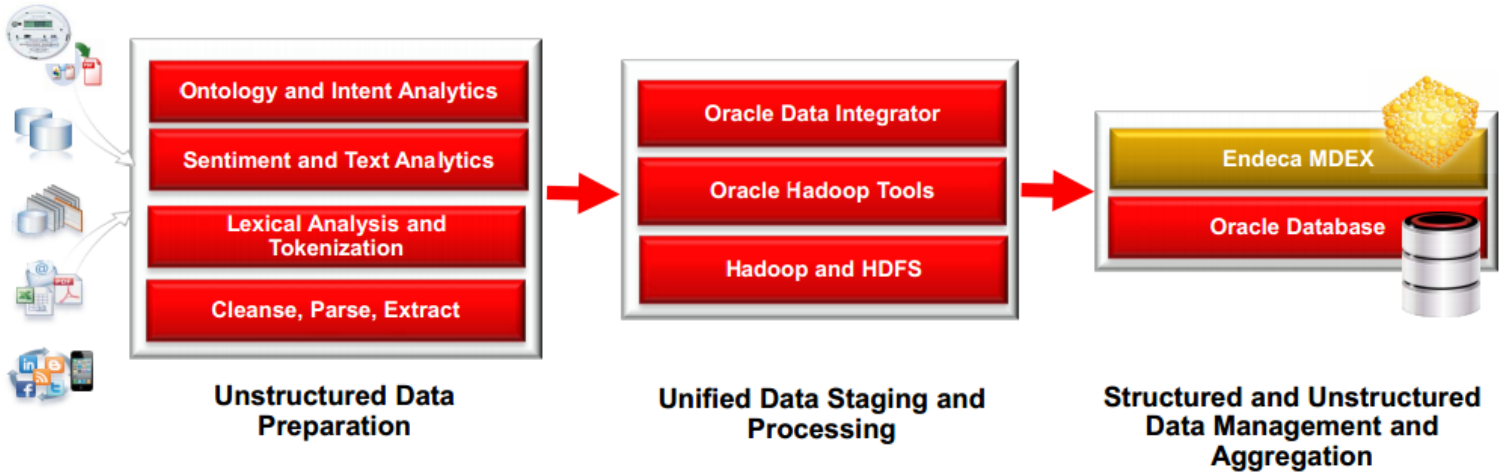
Nous avons vu les langages d'interrogation de données, ils peuvent être utilisés via je JDK ou en ligne de commande, rien de bien beau. Le rêve serait d'avoir un outil plus graphique pour éditer les requêtes.

Un outil open source peut s'occuper de ceci, et apporter un environnement graphique pour gérer ses fichiers, éditer et gérer ses requêtes et tout et tout. Cet outil se nomme **HUE**, voici un petit aperçu:



Et nos chers éditeurs dans tout ca?
La majorité des éditeurs proposent des solutions qui gravitent autour du noyau Hadoop ou d'un noyau technologiquement similaire.
Le schéma ci-dessous représentent une répartition de solutions par domaines:

Endeca est un produit de recherche et d'extraction de données se basant sur l'indexation des fichiers vendu par Oracle. Ce n'est en aucun cas un produit de traitement de données non structurées. Il n'est qu'un outil permettant l'accès à ces données.
Le schéma ci-dessous permet de placer Endeca dans la chaine:



SOURCES:

Map Reduce
http://fr.wikipedia.org/wiki/MapReduce

PIG et HIVE
http://en.wikipedia.org/wiki/Apache_Hive
http://en.wikipedia.org/wiki/Pig_(programming_language)

HUE
http://cloudera.github.io/hue/#home

Endeca
http://www.oracle.com/us/corporate/acquisitions/endeca/general-presentation-517133.pdf