

2019



Big Data, Cloud Computing et Services Web

SAMIR TALBI

NIZAR BEN MHENNI

NIZAR BEN MHENNI;SAMIR TALBI;TALAL ABOU ASSAF

université
de **TOURS**

Contenu

Introduction.....	3
Comment implémenter le projet sur Hadoop :	4
Déployer les tables avec Hbase	4
Déploiement des jobs.....	5
Déploiement de l'ApiRest.jar	5
Récapitulatif APIRest	6
Résultat d'exécution de chaque requête	7
Requête 1 :	7
Résultat API REST:	7
Résultat sur Hbase :	8
Requête 2 :	8
Résultat API REST :	8
Résultat sur Hbase :	9
Requête 3 :	9
Résultat API REST:	9
Résultat sur Hbase :	10
Requête 4 :	10
Résultat API REST:	10
Résultat sur Hbase :	11
Requête 5 :	11
Résultat API REST:	11
Résultat sur Hbase :	12
Requête 6 :	12
Résultat API REST:	12
Résultat sur Hbase :	13
Requête 7 :	13
Résultat API REST:	13
Résultat sur Hbase :	14
Explication de Map Reduce de chaque Job	14
Les Requêtes.....	14
Pour tous les Jobs	14
Requête 1	15
Mapper	15
Reducer.....	16
Requête 2	16

Mapper	16
Reducer.....	16
Requête 3	17
Mapper	17
Reducer.....	17
Requête 4	17
Mapper	17
Reducer.....	18
Methode foundname.....	18
Requête 5	19
Mapper	19
Reducer.....	19
Requête 6	19
Mapper	19
Reducer.....	20
Methode getColumnInColumnFamily	20
Requête 7	21
Mapper	21
Reducer.....	21
Statistique de chaque Job	22
Job1	22
Job2	22
Job3	23
Job4	23
Job5	24
Job6	24
Job7	25

Introduction

Le département d'informatique de l'Université de Blois propose des formations de Licence 1 (L1) à Master 2 (M2) en informatique. Depuis sa création en 2001, chaque promotion possède 200 000 étudiants par an (dont 1 000 000 étudiants chaque année). À chaque semestre, chaque étudiant doit s'inscrire aux 100 UE obligatoires (dont 1 000 UE obligatoires au total pour 10 semestres de L1 à M2) et 100 UE optionnelles parmi les 500 UE au choix (dont 5 000 UE optionnelles au total pour tous les 10 semestres). Les informations ci-dessous sont considérées à des étudiants et à des UE lors de la conception du schéma de base de données.

• Étudiant

- Numéro d'étudiant : l'identification unique d'étudiant composée de 10 chiffres dont 4 pour l'année et 6 pour un nombre formaté, par exemple 2010000001.
- Promotion : la promotion courante d'étudiant, par exemple, L1, M1, etc.
- Nom : le nom d'étudiant, par exemple Dupond.
- Prénom : le prénom d'étudiant, par exemple, Jean.
- Date de naissance : la date de naissance d'étudiant, par exemple, 01/22/1990.
- Email : l'adresse messagerie unique d'étudiant.
- Numéro de téléphone : le numéro de téléphone d'étudiant composé des 10 chiffres.
- Adresse : l'adresse domicile d'étudiant.

• UE

- Code d'UE : l'identification unique d'UE composée des caractères pour le code de semestre, le type et le numéro d'UE dont les UE obligatoires sont identifiées de A001 à A100 et les UE optionnelles sont identifiées de B001 à B500, par exemple S03A001, S10B450, etc.
- Nom d'UE : le nom d'UE qui peut changer selon des années, par exemple UE S07A085 correspondait au cours Compilation entre 2010 et 2013 mais correspond au cours Génération de code depuis 2014.
- Enseignants : la liste d'enseignants qui assurent l'UE

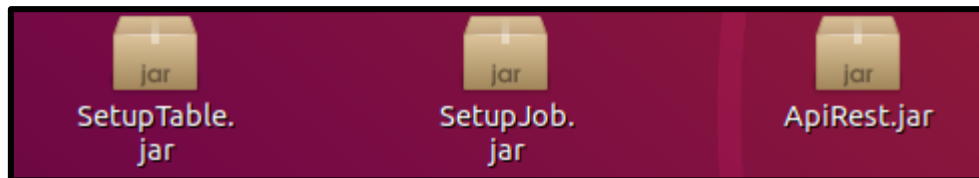
Comment implémenter le projet sur Hadoop :

Lien pour télécharger le projet :

<https://drive.google.com/open?id=1V7YbCX19U0E86E6O0bIk12bGrwXYI8Xo>

Déployer les tables avec Hbase

Pour commencer, apres avoir lancé Hadoop, on telecharge le projet qui se trouve sur OneDrive puis on extrait les tres fichiers neccessaire qu'on a besoin pour executer le projet :



-SetupTable.jar contient les tables student de Hbase

-SetupJob.jar contient les jobs à exécuter

-ApiRest.jar

On execute dans la ligne de commandes nos setup afin de ajouter les tables sur Hbase avec notre namespace qui est **21301866**.

```
public class Namespace {  
  
    private static final String ID = "21301866"; // TODO  
    private static final String courseFamilies = "# I";  
    private static final String courseTable = "C";  
    private static final String gradeFamilies = "#";  
    private static final String gradeTable = "G";  
    private static final String instructorFamilies = "#";  
    private static final String instructorTable = "I";  
    private static final String studentFamilies = "# C";  
    private static final String studentTable = "S";  
}
```

Figure 1 Contenu du Namespace.java

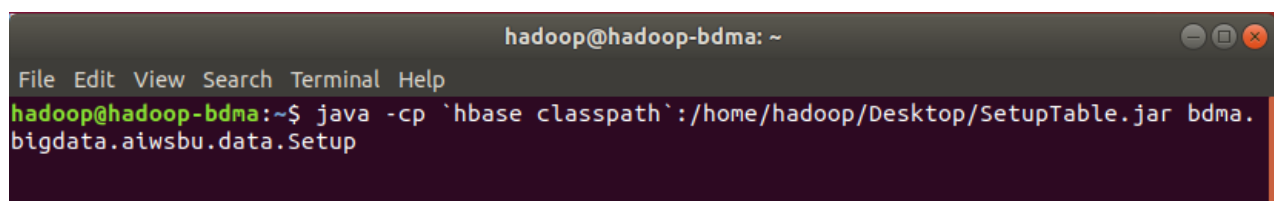


Figure 2 exécution de la commande hbase

```

2019-01-17 18:15:04,333 INFO [main-SendThread(localhost:2181)] zookeeper.Client
Cnxn: Socket connection established to localhost/127.0.0.1:2181, initiating sess
ion
2019-01-17 18:15:04,354 INFO [main-SendThread(localhost:2181)] zookeeper.Client
Cnxn: Session establishment complete on server localhost/127.0.0.1:2181, session
id = 0x1685ccc7d050006, negotiated timeout = 90000
2019-01-17 18:15:06,069 INFO [main] client.HBaseAdmin: Started disable of A:C
2019-01-17 18:15:09,211 INFO [main] client.HBaseAdmin: Created A:C
Table created: A:C
2019-01-17 18:15:09,212 INFO [main] client.HBaseAdmin: Started disable of A:G
2019-01-17 18:15:10,946 INFO [main] client.HBaseAdmin: Created A:G
Table created: A:G
2019-01-17 18:15:10,949 INFO [main] client.HBaseAdmin: Started disable of A:I
2019-01-17 18:15:13,671 INFO [main] client.HBaseAdmin: Created A:I
Table created: A:I
2019-01-17 18:15:13,673 INFO [main] client.HBaseAdmin: Started disable of A:S
2019-01-17 18:15:16,366 INFO [main] client.HBaseAdmin: Created A:S
Table created: A:S
Inserting rows to table: C
Inserting rows to table: S
Inserting rows to table: I
Inserting rows to table: G

```

Après exécuter la commande, les tables seront créées (la table G prend beaucoup temps pour insérer les données).

Déploiement des jobs

Pour déployer les Jobs faits on exécute la commande suivant :

```

hadoop@hadoop-bdma: ~
File Edit View Search Terminal Help
hadoop@hadoop-bdma:~$ java -cp `hbase classpath`:/home/hadoop/Desktop/SetupJob.jar setup.SetupJob

```

Déploiement de l'ApiRest.jar

Pour exécuter l'ApiRest on a besoin de la commande suivante et on laisse le terminal ouvert

```

hadoop@hadoop-bdma: ~/Desktop
File Edit View Search Terminal Help
^Chadoop@hadoop-bdma:~/Desktop$ java -jar ApiRest.jar
log4j:WARN No appenders could be found for logger (org.springframework.web.conte
xt.support.StandardServletEnvironment).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more in
fo.

  ____ _
 / ___ \ | |
/ /___\ \| |
\_____/___|_|
:: Spring Boot :: (v2.0.5.RELEASE)

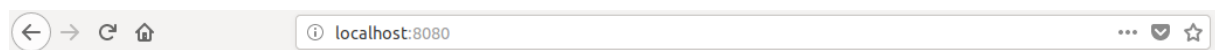
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.springframework.cglib.core.ReflectUtil
s$1 (jar:file:/home/hadoop/Desktop/ApiRest.jar!/BOOT-INF/lib/spring-core-5.0.9.R
ELEASE.jar!) to method java.lang.ClassLoader.defineClass(java.lang.String,byte[
],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of org.springframewor
k.cglib.core.ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflect
ive access operations
WARNING: All illegal access operations will be denied in a future release
Jan 17, 2019 10:14:55 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service [Tomcat]
Jan 17, 2019 10:14:55 PM org.apache.catalina.core.StandardEngine startInternal

```

Après que l'ApiRest est exécuté, on accède à partir d'un navigateur au localhost :

<http://localhost:8080/>

Puis sa nous doit afficher cette page :



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Thu Jan 17 23:29:17 CET 2019

There was an unexpected error (type=Not Found, status=404).

No message available

Récapitulatif API Rest

Afin de mettre nos requêtes sur navigateur, nous avons utilisé une sous structure de Maven qui tourne sous Spring. Nous avons commencé par générer un projet simple qui permet d'avoir une page simple qui contient une simple phrase, ensuite nous avons essayé d'adapter cette structure à Hbase, il aura fallu ajouter les dépendances nécessaires à l'exécution de Hbase. Enfin nous avons codé un contrôleur qui permet le routage de notre API en fonction d'URL entré par l'utilisateur ou la machine afin de récupérer un JSON.

Exemple : query4

```
//QUERY 4
public static String getquery4(String id,String year) throws IOException {
    HTable table = new HTable(conf, "21301866:query4");
    String reponse;
    if(id.length()==7 && year.length()==4) {
        Get get = new Get(Bytes.toBytes(id+"/"+year));
        Result result = table.get(get);
        if(Bytes.toString(result.getValue(Bytes.toBytes("#"), Bytes.toBytes("N")))==null) {
            reponse = "NOT FOUND";
        }else{
            byte[] name = result.getValue(Bytes.toBytes("#"), Bytes.toBytes("N"));
            byte[] taux = result.getValue(Bytes.toBytes("#"), Bytes.toBytes("NT"));
            reponse = "{\"Name\":\""+Bytes.toString(name)+"\", \"Rate\":\""+ Bytes.toString(taux)+"\"}";
        }
    }else {
        return "FORMAT INVALID (id=S00A000 year=2000)";
    }
    return reponse;
}
```

Récupération de routage pour la question 4 où nous renvoyons les données générées en format JSON de la méthode GetQuery4 en fonction de deux attributs récupéré dans le mapping.

Exemple :

```
//http://localhost:8080/courses/S07A010/rates/2010
@RequestMapping(value = "/courses/{id}/rates/{year}", method = RequestMethod.GET, produces = "application/json")
public Object query4(@PathVariable("id") String id, @PathVariable("year") String year) throws IOException {
    return getquery4(id,year);
}
```

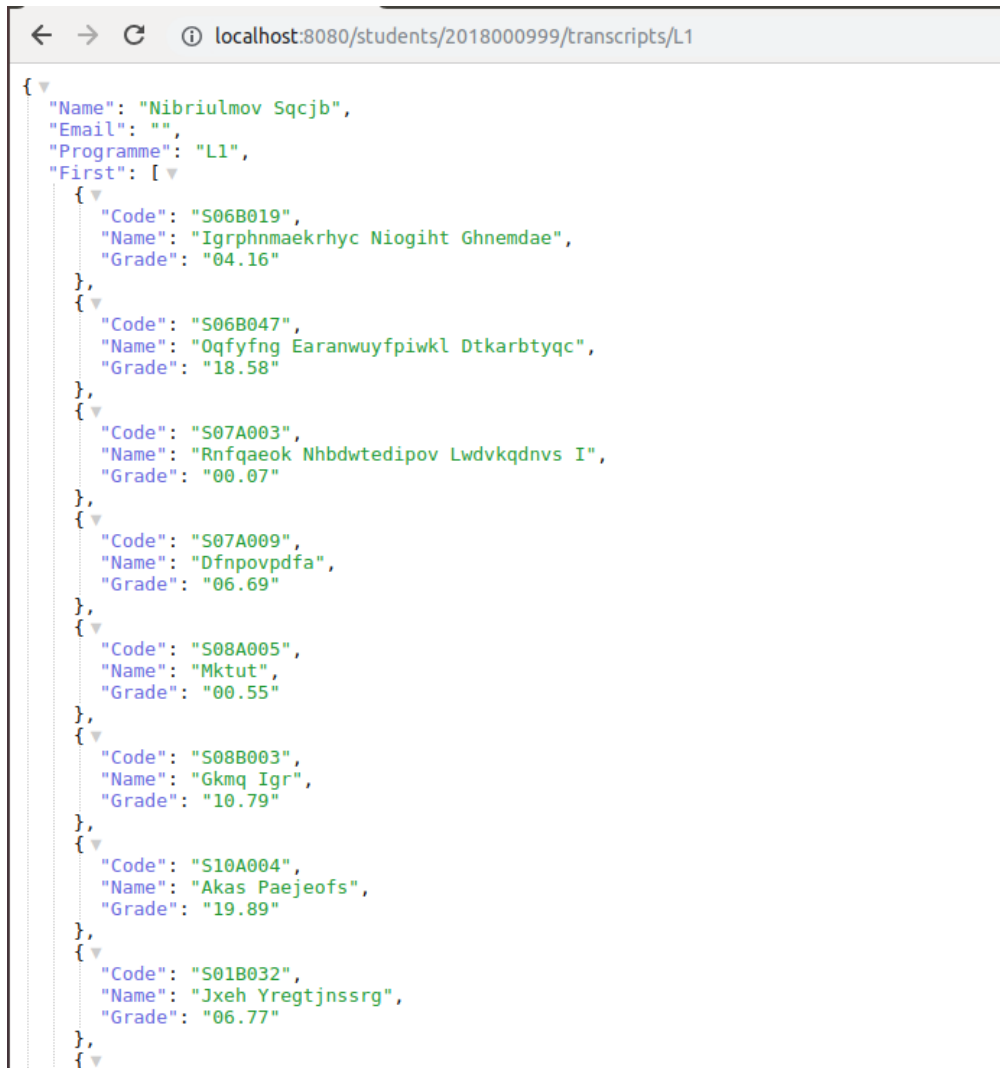
Résultat d'exécution de chaque requête

Requête 1 :

Question : imprimer les relevés de notes par année scolaire.

Lien : <http://localhost:8080/students/2018000999/transcripts/L1>

Résultat API REST:



```
{
  "Name": "Nibriulmov Sqcjb",
  "Email": "",
  "Programme": "L1",
  "First": [
    {
      "Code": "S06B019",
      "Name": "Igrphnmaekrhyc Niogiht Ghnemdae",
      "Grade": "04.16"
    },
    {
      "Code": "S06B047",
      "Name": "Oqfyfng Earanwuyfpiwkl Dtkarbtyqc",
      "Grade": "18.58"
    },
    {
      "Code": "S07A003",
      "Name": "Rnfqaeok Nhbdwtedipov Lwdvkqdnvs I",
      "Grade": "00.07"
    },
    {
      "Code": "S07A009",
      "Name": "Dfnpovpdfa",
      "Grade": "06.69"
    },
    {
      "Code": "S08A005",
      "Name": "Mktut",
      "Grade": "00.55"
    },
    {
      "Code": "S08B003",
      "Name": "Gkmq Igr",
      "Grade": "10.79"
    },
    {
      "Code": "S10A004",
      "Name": "Akas Paejeofs",
      "Grade": "19.89"
    },
    {
      "Code": "S01B032",
      "Name": "Jxeh Yregtjnssrg",
      "Grade": "06.77"
    }
  ]
}
```


Résultat sur Hbase :

```
2018001000/L1 column=:FS, timestamp=1548187696341, value=S10B033/W Cnaa
qhs So/1885;S10B023/Sqx Lprymrrljnl/0203;S10B009/Plapmkhf
inm/1327;S10A010/C/0174;S10A004/Akas Paejeofs/1040;S10A001
/Ytt Klsjfxmca/0941;S09B032/Yhocwr Djubpxdmylryp/0423;S09B
003/Me Ne/0088;S09A001/Umfegkgqjpi Aruoijug/0416;S08B047/W
ny Cg Bbxvbqdwqfakqb Hwbybebgqtfe/1466;S08A005/Mktut/1824;
S07B026/Wukegmbyjthpxm Hkliggvxncmgn/0776;S07A006/Mrlosfq
Qrcch/0448;S06B021/Kogwjtvqrhrb T Jfr/1608;S05B019/Ydagksn
odjcbwm/0744;S05A005/Jop L Gfa/0657;S04A004/Agoo Qpdcdwgox
Aneysol/1505;S03B006/Ftavs Qdmsbllds/0469;S03A001/Hqiokjo
ynrtwvy Tpmk/0205;S02A007/Iceckpac Xcfaaa Iqfltdu/0880
2018001000/L1 column=:SS, timestamp=1548187696341, value=S04B017/Vmxq/1
890;S03B039/Jueo Bejwcnn/0581;S03A010/Vjcqqksvmsmg Bclcvkc
jr/1970;S01B036/Oqcjmd Oauljbca Lsixem/1475;S01A008/Knmadm
ekbepeq/1073;S10B002/Qgyki Icinky/1519;S10A004/Akas Paejeo
fs/1168;S10A001/Ytt Klsjfxmca/1962;S09A006/Vhpjrlygbw/1277
;S08B022/Itkcq H/1955;S07B050/Gkvjoywnppwxww Akwwy/1162;S0
7B008/Egbmxjvg Mdlmseplbgmmwo Mjypfnomb/1376;S06B026/Iwyup
ncndi Djslpmtv K Fd/0041;S06B021/Kogwjtvqrhrb T Jfr/0882;S
06A010/Cnb Cfpnnilcvckhm Csaqkvgyfyk Rjonky/0926;S06A002/Y
unbbdqcln/1345;S05B022/Nvnesoxaehg Xtbl/0879;S05A009/Ihdc/
0440;S05A008/Jxkmwvxvdwvdk Pi/0636;S05A007/Vflx/0138
42529 row(s) in 42.7260 seconds
```

Requête 2 :

Question : Savoir le taux de réussite d'un semestre (de 1 à 10) selon l'année scolaire.

Lien : <http://localhost:8080/rates/03>

Résultat API REST :

localhost:8080/rates/03	
JSON	Raw Data Headers
Save	Copy
Filter JSON	
0:	Year: "2004"
	Rate: "0.5"
1:	Year: "2005"
	Rate: "0.5"
2:	Year: "2006"
	Rate: "0.5"
3:	Year: "2007"
	Rate: "0.5"
4:	Year: "2008"
	Rate: "0.5"
5:	Year: "2009"
	Rate: "0.5"
6:	Year: "2010"
	Rate: "0.51"
7:	Year: "2011"
	Rate: "0.51"
8:	Year: "2012"
	Rate: "0.5"
9:	Year: "2013"
	Rate: "0.5"

Résultat sur Hbase :

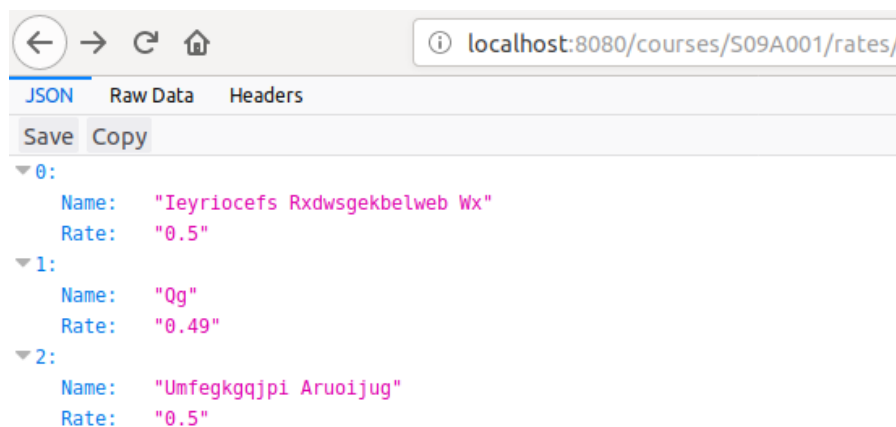
```
2018/04 column=:TR, timestamp=1548188318306, value=0.5
2018/05 column=:TR, timestamp=1548188318306, value=0.5
2018/06 column=:TR, timestamp=1548188318306, value=0.49
2018/07 column=:TR, timestamp=1548188318306, value=0.5
2018/08 column=:TR, timestamp=1548188318306, value=0.49
2019/01 column=:TR, timestamp=1548188318306, value=0.5
2019/02 column=:TR, timestamp=1548188318306, value=0.5
2019/03 column=:TR, timestamp=1548188318306, value=0.5
2019/04 column=:TR, timestamp=1548188318306, value=0.5
2019/05 column=:TR, timestamp=1548188318306, value=0.5
2019/06 column=:TR, timestamp=1548188318306, value=0.5
2019/07 column=:TR, timestamp=1548188318306, value=0.5
2019/08 column=:TR, timestamp=1548188318306, value=0.51
2020/03 column=:TR, timestamp=1548188318306, value=0.5
2020/04 column=:TR, timestamp=1548188318306, value=0.5
2020/05 column=:TR, timestamp=1548188318306, value=0.5
2020/06 column=:TR, timestamp=1548188318306, value=0.5
2020/07 column=:TR, timestamp=1548188318306, value=0.5
2020/08 column=:TR, timestamp=1548188318306, value=0.5
2021/05 column=:TR, timestamp=1548188318306, value=0.5
2021/06 column=:TR, timestamp=1548188318306, value=0.5
2021/07 column=:TR, timestamp=1548188318306, value=0.5
2021/08 column=:TR, timestamp=1548188318306, value=0.5
2022/05 column=:TR, timestamp=1548188318306, value=0.51
2022/06 column=:TR, timestamp=1548188318306, value=0.5
2022/07 column=:TR, timestamp=1548188318306, value=0.49
2022/08 column=:TR, timestamp=1548188318306, value=0.5
2023/07 column=:TR, timestamp=1548188318306, value=0.49
2023/08 column=:TR, timestamp=1548188318306, value=0.49
2024/07 column=:TR, timestamp=1548188318306, value=0.51
2024/08 column=:TR, timestamp=1548188318306, value=0.5
2025/07 column=:TR, timestamp=1548188318306, value=0.51
2025/08 column=:TR, timestamp=1548188318306, value=0.49
132 row(s) in 0.0630 seconds
```

Requête 3 :

Question : Savoir le taux de réussite d'une UE depuis sa création, par rapport, au cas où, à ses différents noms.

Lien : <http://localhost:8080/courses/S09A001/rates>

Résultat API REST:



The screenshot shows a web browser window with the address bar displaying `localhost:8080/courses/S09A001/rates/`. The browser's developer tools or a dedicated API client interface is open, showing the response in JSON format. The JSON array contains three objects, each with a 'Name' and a 'Rate' field.

```

{
  "0": {
    "Name": "Ieyriocefs Rxdwsgekelweb Wx",
    "Rate": "0.5"
  },
  "1": {
    "Name": "Qg",
    "Rate": "0.49"
  },
  "2": {
    "Name": "Umfegkgqjpi Aruoijug",
    "Rate": "0.5"
  }
}

```

Résultat sur Hbase :

```
S10B027/Dtfwxy Tuouyh Inehuuvvcfhc column=#:TR, timestamp=1548188454358, value=0.51
S10B028/Je Klxgfjpe column=#:TR, timestamp=1548188454358, value=0.51
S10B030/Hkednlpn Ykjcmo column=#:TR, timestamp=1548188454358, value=0.5
S10B031/Jjp Jumwhh J column=#:TR, timestamp=1548188454358, value=0.51
S10B031/Mt Nth column=#:TR, timestamp=1548188454358, value=0.51
S10B032/Puxvckstmyvf Yjmcnitfsk column=#:TR, timestamp=1548188454358, value=0.5
S10B032/Vrpfmydgke Uuuosvcwe Mtv Bqtn column=#:TR, timestamp=1548188454358, value=0.5
oaoqw
S10B033/JfqxarfD Bcjitow Muv Xdckelb column=#:TR, timestamp=1548188454358, value=0.49
S10B033/W Cnaaqhs So column=#:TR, timestamp=1548188454358, value=0.49
S10B034/Nvcbrnkiinwi Gwbylpdfmf column=#:TR, timestamp=1548188454358, value=0.49
S10B034/Ypxfirup Aufochr column=#:TR, timestamp=1548188454358, value=0.5
S10B035/Fcxb I M column=#:TR, timestamp=1548188454358, value=0.51
S10B035/Vvxattowolsf Hxkfoysx column=#:TR, timestamp=1548188454358, value=0.49
S10B036/Btfsn column=#:TR, timestamp=1548188454358, value=0.48
S10B036/Vhdet G Proshoub column=#:TR, timestamp=1548188454358, value=0.47
S10B037/Eu Aa column=#:TR, timestamp=1548188454358, value=0.5
S10B038/Dutxjnecvicio column=#:TR, timestamp=1548188454358, value=0.51
S10B038/Jiqxfy Vygehfwqpldh column=#:TR, timestamp=1548188454358, value=0.54
S10B038/Nwhrqqbudg Mnlv column=#:TR, timestamp=1548188454358, value=0.5
S10B039/Nrxrlqe column=#:TR, timestamp=1548188454358, value=0.48
S10B040/N Npqkkgfmguhr Xoljqihebswu column=#:TR, timestamp=1548188454358, value=0.47
S10B041/Fimroci Wbw Pj column=#:TR, timestamp=1548188454358, value=0.5
S10B042/Blwwlndsper Cvfxllmtvdoy Qky column=#:TR, timestamp=1548188454358, value=0.49
xhdaq Mx
S10B042/H Saibxhvbgrnt column=#:TR, timestamp=1548188454358, value=0.48
S10B043/Brr Hyg Nglguf column=#:TR, timestamp=1548188454358, value=0.49
S10B043/Hh column=#:TR, timestamp=1548188454358, value=0.52
S10B044/Utxok column=#:TR, timestamp=1548188454358, value=0.51
S10B045/Jjrtiserjifxl Gk Wecrvel column=#:TR, timestamp=1548188454358, value=0.5
S10B045/Xshypfqyeuc Wisuadiu column=#:TR, timestamp=1548188454358, value=0.5
S10B046/Osrhfpglybqki Mvrl Hvbdnx column=#:TR, timestamp=1548188454358, value=0.51
S10B046/Vbri Wf E column=#:TR, timestamp=1548188454358, value=0.49
S10B048/Xsjlegusu Mofgnkfrqg column=#:TR, timestamp=1548188454358, value=0.5
S10B049/Dubwfh Wtcjag column=#:TR, timestamp=1548188454358, value=0.52
S10B049/Ed Gtssb column=#:TR, timestamp=1548188454358, value=0.49
S10B049/Gcoeeapx Mipdddn column=#:TR, timestamp=1548188454358, value=0.5
900 row(s) in 0.1630 seconds
```

Requête 4 :

Question : Plus précisément, savoir le taux de réussite d'une UE pour une année scolaire.

Lien : <http://localhost:8080/courses/S09A001/rates/2011>

Résultat API REST:

←

→

↺

🏠

localhost:8080/courses/S09A001/rates/2011

JSON

Raw Data

Headers

Save

Copy

Name: "Ieyriocefs Rxdwsgekelweb Wx"

Rate: "0.48"

Résultat sur Hbase :

```

S10B049/2016      column=#:NT, timestamp=1548188682379, value=0.45
S10B049/2016      column=#:T, timestamp=1548188682379, value=187
S10B049/2017      column=#:C, timestamp=1548188682379, value=88
S10B049/2017      column=#:N, timestamp=1548188682379, value=Ed Gtssb
S10B049/2017      column=#:NT, timestamp=1548188682379, value=0.48
S10B049/2017      column=#:T, timestamp=1548188682379, value=185
S10B049/2018      column=#:C, timestamp=1548188682379, value=87
S10B049/2018      column=#:N, timestamp=1548188682379, value=Ed Gtssb
S10B049/2018      column=#:NT, timestamp=1548188682379, value=0.48
S10B049/2018      column=#:T, timestamp=1548188682379, value=180
S10B049/2019      column=#:C, timestamp=1548188682379, value=94
S10B049/2019      column=#:N, timestamp=1548188682379, value=Ed Gtssb
S10B049/2019      column=#:NT, timestamp=1548188682379, value=0.47
S10B049/2019      column=#:T, timestamp=1548188682379, value=199
S10B049/2020      column=#:C, timestamp=1548188682379, value=65
S10B049/2020      column=#:N, timestamp=1548188682379, value=Ed Gtssb
S10B049/2020      column=#:NT, timestamp=1548188682379, value=0.53
S10B049/2020      column=#:T, timestamp=1548188682379, value=123
S10B049/2021      column=#:C, timestamp=1548188682379, value=25
S10B049/2021      column=#:N, timestamp=1548188682379, value=Ed Gtssb
S10B049/2021      column=#:NT, timestamp=1548188682379, value=0.45
S10B049/2021      column=#:T, timestamp=1548188682379, value=55
S10B049/2022      column=#:C, timestamp=1548188682379, value=27
S10B049/2022      column=#:N, timestamp=1548188682379, value=Ed Gtssb
S10B049/2022      column=#:NT, timestamp=1548188682379, value=0.48
S10B049/2022      column=#:T, timestamp=1548188682379, value=56
S10B049/2023      column=#:C, timestamp=1548188682379, value=9
S10B049/2023      column=#:N, timestamp=1548188682379, value=Ed Gtssb
S10B049/2023      column=#:NT, timestamp=1548188682379, value=0.6
S10B049/2023      column=#:T, timestamp=1548188682379, value=15
S10B049/2024      column=#:C, timestamp=1548188682379, value=9
S10B049/2024      column=#:N, timestamp=1548188682379, value=Ed Gtssb
S10B049/2024      column=#:NT, timestamp=1548188682379, value=0.5
S10B049/2024      column=#:T, timestamp=1548188682379, value=18
S10B049/2025      column=#:C, timestamp=1548188682379, value=11
S10B049/2025      column=#:N, timestamp=1548188682379, value=Ed Gtssb
S10B049/2025      column=#:NT, timestamp=1548188682379, value=0.52
S10B049/2025      column=#:T, timestamp=1548188682379, value=21
11903 row(s) in 4.5780 seconds

```

Requête 5 :

Question : imprimer les relevés de notes par année scolaire.

Lien : <http://localhost:8080/programs/L1/means/2010>

Résultat API REST:

```

← → ↺ ① localhost:8080/programs/L1/means/2010
{
  "S01A001": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Grade": "9.88"
  },
  "S01A002": {
    "Name": "Qi V H",
    "Grade": "10.06"
  },
  "S01A003": {
    "Name": "Fdv gjapvalrs Yeu",
    "Grade": "10.1"
  },
  "S01A004": {
    "Name": "Rhelxv Spccg",
    "Grade": "9.64"
  },
  "S01A005": {
    "Name": "Nnqocwkuw Tnuusr",
    "Grade": "10.34"
  },
  "S01A007": {
    "Name": "Sfowuyyxuexwi Kuplt",
    "Grade": "10.1"
  }
}

```


Résultat sur Hbase :

```
S10B049/2017/M1 column=#:M, timestamp=1548189183134, value=9.033571
S10B049/2017/M1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2018/L1 column=#:M, timestamp=1548189183134, value=10.25
S10B049/2018/L1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2018/L2 column=#:M, timestamp=1548189183134, value=9.688923
S10B049/2018/L2 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2018/L3 column=#:M, timestamp=1548189183134, value=7.7600007
S10B049/2018/L3 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2018/M1 column=#:M, timestamp=1548189183134, value=12.33222
S10B049/2018/M1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2019/L1 column=#:M, timestamp=1548189183134, value=9.7216
S10B049/2019/L1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2019/L2 column=#:M, timestamp=1548189183134, value=9.045396
S10B049/2019/L2 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2019/L3 column=#:M, timestamp=1548189183134, value=9.914186
S10B049/2019/L3 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2019/M1 column=#:M, timestamp=1548189183134, value=9.641666
S10B049/2019/M1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2020/L2 column=#:M, timestamp=1548189183134, value=10.118067
S10B049/2020/L2 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2020/L3 column=#:M, timestamp=1548189183134, value=9.92111
S10B049/2020/L3 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2020/M1 column=#:M, timestamp=1548189183134, value=11.7144
S10B049/2020/M1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2021/L3 column=#:M, timestamp=1548189183134, value=8.86878
S10B049/2021/L3 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2021/M1 column=#:M, timestamp=1548189183134, value=12.812143
S10B049/2021/M1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2022/L3 column=#:M, timestamp=1548189183134, value=10.949428
S10B049/2022/L3 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2022/M1 column=#:M, timestamp=1548189183134, value=8.249048
S10B049/2022/M1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2023/M1 column=#:M, timestamp=1548189183134, value=10.700001
S10B049/2023/M1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2024/M1 column=#:M, timestamp=1548189183134, value=9.991666
S10B049/2024/M1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
S10B049/2025/M1 column=#:M, timestamp=1548189183134, value=11.508572
S10B049/2025/M1 column=#:N, timestamp=1548189183134, value=Ed Gtssb
32732 row(s) in 6.7730 seconds
```

Requête 6 :

Question : savoir les taux de réussite de toutes les UE assurées par un intervenant.

Lien : <http://localhost:8080/instructors/Qu%20Owtd/rates>

Résultat API REST:

```
← → ↺ ⓘ localhost:8080/instructors/Qu%20Owtd/rates
{
  "S01A001/2002": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Rate": ""
  },
  "S01A001/2003": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Rate": "0.45"
  },
  "S01A001/2004": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Rate": "0.52"
  },
  "S01A001/2005": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Rate": "0.48"
  },
  "S01A001/2006": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Rate": "0.45"
  },
  "S01A001/2007": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Rate": "0.48"
  },
  "S01A001/2008": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Rate": "0.5"
  },
  "S01A001/2009": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Rate": "0.47"
  },
  "S01A001/2010": {
    "Name": "Vybfhdjeckje Ayitbyavmger Ttaoxr",
    "Rate": ""
  }
}
```

Résultat sur Hbase :

```
S10B049/2020/Sptwckematrak Ayb column=:TR, timestamp=1548189395410, value=0.53
S10B049/2021/Bprv Bscgn column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2021/Bprv Bscgn column=:TR, timestamp=1548189395410, value=0.45
S10B049/2021/Qu Owtdxug column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2021/Qu Owtdxug column=:TR, timestamp=1548189395410, value=0.45
S10B049/2021/Sptwckematrak Ayb column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2021/Sptwckematrak Ayb column=:TR, timestamp=1548189395410, value=0.45
S10B049/2022/Bprv Bscgn column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2022/Bprv Bscgn column=:TR, timestamp=1548189395410, value=0.48
S10B049/2022/Qu Owtdxug column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2022/Qu Owtdxug column=:TR, timestamp=1548189395410, value=0.48
S10B049/2022/Sptwckematrak Ayb column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2022/Sptwckematrak Ayb column=:TR, timestamp=1548189395410, value=0.48
S10B049/2023/Bprv Bscgn column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2023/Bprv Bscgn column=:TR, timestamp=1548189395410, value=0.6
S10B049/2023/Qu Owtdxug column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2023/Qu Owtdxug column=:TR, timestamp=1548189395410, value=0.6
S10B049/2023/Sptwckematrak Ayb column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2023/Sptwckematrak Ayb column=:TR, timestamp=1548189395410, value=0.6
S10B049/2024/Bprv Bscgn column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2024/Bprv Bscgn column=:TR, timestamp=1548189395410, value=0.5
S10B049/2024/Qu Owtdxug column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2024/Qu Owtdxug column=:TR, timestamp=1548189395410, value=0.5
S10B049/2024/Sptwckematrak Ayb column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2024/Sptwckematrak Ayb column=:TR, timestamp=1548189395410, value=0.5
S10B049/2025/Bprv Bscgn column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2025/Bprv Bscgn column=:TR, timestamp=1548189395410, value=0.52
S10B049/2025/Qu Owtdxug column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2025/Qu Owtdxug column=:TR, timestamp=1548189395410, value=0.52
S10B049/2025/Sptwckematrak Ayb column=:N, timestamp=1548189395410, value=Ed Gtssb
S10B049/2025/Sptwckematrak Ayb column=:TR, timestamp=1548189395410, value=0.52
23872 row(s) in 5.5040 seconds
```

Requête 7 :

Question : savoir le classement d'étudiants par rapport à leurs notes moyennes selon la promotion et l'année scolaire.

Lien : <http://localhost:8080/ranks/L1/years/2010>

Résultat API REST:

```
← → ↺ ⓘ localhost:8080/ranks/L1/years/2010

{
  "2009000250": "12.82",
  "2009000430": "12.45",
  "2009000072": "12.38",
  "2009000404": "12.36",
  "2009000273": "12.30",
  "2009000738": "12.30",
  "2009000382": "12.28",
  "2009000802": "12.25",
  "2009000795": "12.20",
  "2009000231": "12.18",
  "2009000994": "12.16",
  "2009000700": "12.15",
  "2009000823": "12.15",
  "2009000358": "12.14",
  "2009000138": "12.13",
  "2009000318": "12.12",
  "2009000806": "12.10",
  "2009000410": "12.04",
  "2009000017": "12.03",
  "2009000192": "12.00",
  "2009000526": "12.00",
  "2009000603": "12.00",
  "2009000418": "11.92",
  "2009000730": "11.92",
  "2009000578": "11.90",
  "2009000596": "11.90",
  "2009000783": "11.90",
  "2009000432": "11.88",
  "2009000299": "11.86",
  "2009000443": "11.86",
  "2009000886": "11.84",
  "2009000866": "11.83",
  "2009000676": "11.81",
}
```

Résultat sur Hbase :

```

1289/2008000045/2014/L3      column=:M, timestamp=1548189464599, value=07.11
1289/20090000842/2010/L1      column=:M, timestamp=1548189464599, value=07.11
1289/20130000825/2019/L3      column=:M, timestamp=1548189464926, value=07.11
1289/20140000448/2020/L3      column=:M, timestamp=1548189465102, value=07.11
1291/20110000482/2012/L1      column=:M, timestamp=1548189464752, value=07.09
1291/20150000142/2018/L2      column=:M, timestamp=1548189465102, value=07.09
1292/20150000108/2016/L1      column=:M, timestamp=1548189465102, value=07.08
1294/20050000751/2008/L2      column=:M, timestamp=1548189464409, value=07.06
1294/20080000200/2011/L2      column=:M, timestamp=1548189464599, value=07.06
1294/20110000734/2017/L3      column=:M, timestamp=1548189464752, value=07.06
1296/20150000434/2021/L3      column=:M, timestamp=1548189465102, value=07.04
1297/20040000169/2005/L1      column=:M, timestamp=1548189464210, value=07.03
1300/20020000574/2003/L1      column=:M, timestamp=1548189464089, value=07.00
1301/20050000797/2006/L1      column=:M, timestamp=1548189464409, value=06.99
1301/20140000963/2024/M1      column=:M, timestamp=1548189465102, value=06.99
1302/20100000048/2011/L1      column=:M, timestamp=1548189464752, value=06.98
1304/20020000373/2003/L1      column=:M, timestamp=1548189464089, value=06.96
1306/20110000808/2012/L1      column=:M, timestamp=1548189464752, value=06.94
1308/20060000405/2012/L3      column=:M, timestamp=1548189464409, value=06.92
1309/20120000489/2018/L3      column=:M, timestamp=1548189464926, value=06.91
1313/20120000410/2015/L2      column=:M, timestamp=1548189464926, value=06.87
1314/20100000483/2016/L3      column=:M, timestamp=1548189464752, value=06.86
1318/20060000677/2012/L3      column=:M, timestamp=1548189464409, value=06.82
1320/20140000834/2015/L1      column=:M, timestamp=1548189465102, value=06.80
1325/20130000172/2016/L2      column=:M, timestamp=1548189464926, value=06.75
1326/20040000436/2010/L3      column=:M, timestamp=1548189464210, value=06.74
1327/20030000331/2004/L1      column=:M, timestamp=1548189464210, value=06.73
1327/20160000676/2017/L1      column=:M, timestamp=1548189465102, value=06.73
1329/20020000445/2005/L2      column=:M, timestamp=1548189464089, value=06.71
1329/20140000792/2015/L1      column=:M, timestamp=1548189465102, value=06.71
1329/20160000325/2017/L1      column=:M, timestamp=1548189465102, value=06.71
1332/20100000526/2016/L3      column=:M, timestamp=1548189464752, value=06.68
1334/20030000220/2009/L3      column=:M, timestamp=1548189464089, value=06.66
1342/20100000801/2013/L2      column=:M, timestamp=1548189464752, value=06.58
1345/20050000971/2006/L1      column=:M, timestamp=1548189464409, value=06.55
1349/20040000921/2010/L3      column=:M, timestamp=1548189464210, value=06.51
1370/20030000596/2004/L1      column=:M, timestamp=1548189464210, value=06.30
1383/20080000949/2011/L2      column=:M, timestamp=1548189464599, value=06.17
42529 row(s) in 4.5360 seconds

```

Explication de Map Reduce de chaque Job

Les Requêtes

Pour tous les Jobs

Dans ce projet, nous avons été amenés à faire plusieurs jobs, ces jobs sont lancés à partir d'une classe qui est presque semblable pour tous les jobs. Celle-ci commence par la désactivation de la table et sa suppression si elle existe et ensuite sa création. Ensuite nous avons une table source (qui correspond à la table où se trouve nos données que nous allons utiliser) et une table target (qui sera la table où nous allons stock nos données réorganisés). Enfin nous faisons appel au mapper et reducer que chaque job.

```

Configuration config = HBaseConfiguration.create();
Job job = new Job(config, "A");
job.setJarByClass(map1.class);
String sourceTable = setting.getId() + ":G";
String targetTable = setting.getId() + ":query1";
HBaseAdmin admin = new HBaseAdmin(config);
if (admin.tableExists(targetTable)) {
    admin.disableTable(targetTable);
    admin.deleteTable(targetTable);
}

HTableDescriptor tableDescriptor = new HTableDescriptor(TableName.valueOf(targetTable));
tableDescriptor.addFamily(new HColumnDescriptor("#"));
admin.createTable(tableDescriptor);
TableMapReduceUtil.initTableMapperJob(sourceTable, new Scan(), map1.class, Text.class, Text.class, job);
TableMapReduceUtil.initTableReducerJob(targetTable, reducer1.class, job);
job.setNumReduceTasks(1);
boolean b = job.waitForCompletion(true);
if (!b) {
    throw new IOException("error with job!");
}

```

Requête 1

Mapper

```

public void map(ImmutableBytesWritable key, Result value, Mapper<ImmutableBytesWritable, Result> context) throws IOException, InterruptedException {
    byte[] values = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("G"));
    String val = Bytes.toString(values);
    String[] key2 = Bytes.toString(key.get()).split("/");
    String prog = "";
    String programme = key2[1].substring(0, 2);
    if (programme.equals("01") || programme.equals("02")) {
        prog = "L1";
    }

    if (programme.equals("03") || programme.equals("04")) {
        prog = "L2";
    }

    if (programme.equals("05") || programme.equals("06")) {
        prog = "L3";
    }

    if (programme.equals("07") || programme.equals("08")) {
        prog = "M1";
    }

    if (programme.equals("09") || programme.equals("10")) {
        prog = "M2";
    }

    String newkey = key2[1].substring(2) + "/" + prog;
    this.text.set(programme + "/" + key2[2] + "/" + val + "/" + key2[0]);
    context.write(new Text(newkey), this.text);
}

```

Dans le mapper de la requête 1, nous lisons la table grade afin d'avoir le programme (que nous transformons depuis le semestre) et les notes pour chaque étudiant, puis nous créons les clés pour chaque row de notre nouvelle table qui se compose en NumeroEtu/programme (par exemple : 200500005/L3) que nous envoyons au reducer avec comme valeur : programme/codeMatiere/note/annee.

Reducer

```
HTable courses = new HTable(conf, setting.getId()+"query4");

ArrayList<String> Fsemestre = new ArrayList<String>();
ArrayList<String> Ssemestre = new ArrayList<String>();

for (Text vals : values) {
    String[] Valtable = vals.toString().split("/");
    Get get = new Get(Bytes.toBytes(Valtable[1]+"/"+Valtable[3]));
    Result resultget = courses.get(get);
    byte [] newnom = resultget.getValue(Bytes.toBytes("#"), Bytes.toBytes("N"));
    String newn = Bytes.toString(newnom);
    if(Valtable[0].equals("01") || Valtable[0].equals("03") || Valtable[0].equals("05") || Valtable[0].equals("07") || Valtable[0].equals("09")) {
        Fsemestre.add(Valtable[1]+"/"+ newn +"/"+ Valtable[2]);
    }
    if(Valtable[0].equals("02") || Valtable[0].equals("04") || Valtable[0].equals("06") || Valtable[0].equals("08") || Valtable[0].equals("10")) {
        Ssemestre.add(Valtable[1]+"/"+ newn +"/"+ Valtable[2]);
    }
}

Put put = new Put (Bytes.toBytes(key.toString()));
put.add(Bytes.toBytes("#"), Bytes.toBytes("FS"), Bytes.toBytes(String.valueOf(String.join(";", Fsemestre))));
put.add(Bytes.toBytes("#"), Bytes.toBytes("SS"), Bytes.toBytes(String.valueOf(String.join(";", Ssemestre))));
context.write(null, put);
```

On a créé un string qui contient code/nameMatiere/value séparer par des points virgule ranger selon le premier et second semestre.

Requête 2

Mapper

```
byte[] values = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("G"));
String val = Bytes.toString(values);
String[] key2 = Bytes.toString(key.get()).split("/");
String newkey = key2[0] + "/" + key2[1].substring(0, 2);
this.text.set(val);
context.write(new Text(newkey), this.text);
```

Dans ce mapper, nous avons lu la table Grade qui nous a permis de récupérer les notes, et on a changé la clé pour chaque valeur afin de correspondre ceci : année/semestre. Ensuite, nous avons envoyé ces valeurs vers le reducer

Reducer

```
float total=0;
int nb = 0;
for (Text vals : values) {
    if(Float.parseFloat(vals.toString())>=1000) {
        nb++;
    }
    total++;
}

float valu = (float) Math.round(((float) nb / total)*100)/100;
Put put = new Put (Bytes.toBytes(key.toString()));
put.add(Bytes.toBytes("#"), Bytes.toBytes("TR"), Bytes.toBytes(String.valueOf(valu)));
context.write(null, put);
```

Dans le reducer, nous regardons le nombre de personne qui a plus de 10 de moyenne et nous calculons le taux de réussite, donc à la fin nous avons une table qui contient comme clé(row) : année/semestre avec le taux de réussite.

Requête 3

Mapper

```
byte[] name = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("N"));
byte[] cumule = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("C"));
byte[] total = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("T"));
String name1 = Bytes.toString(name);
String cumuleval = Bytes.toString(cumule);
String totalval = Bytes.toString(total);
String[] key2 = Bytes.toString(key.get()).split("/");
String newkey = key2[0] + "/" + name1;
this.text.set(cumuleval + "/" + totalval);
context.write(new Text(newkey), this.text);
```

Pour la Job 3, nous avons utilisé la table de la requête 4, car cela est plus simple. Nous avons retravaillé la clé afin de mieux correspondre à nos besoins. La nouvelle clé correspond à codeMatiere/nameMatiere avec comme valeur la somme des valeur / nombre totale de valeur.

Reducer

```
int total = 0;
int nb = 0;

for (Text vals : values) {
    String[] listenote = vals.toString().split("/");
    nb += Integer.parseInt(listenote[0]);
    total += Integer.parseInt(listenote[1]);
}

float valu = (float) Math.round(((float) nb / total)*100)/100;
String key3 = key.toString();

Put put = new Put(Bytes.toBytes(key3));
put.add(Bytes.toBytes("#"), Bytes.toBytes("TR"), Bytes.toBytes(String.valueOf(valu)));

context.write(null, put);
```

Dans le reducer, nous faisons le calcul des taux de réussite selon les différents noms de la matière, puis nous mettons les valeurs dans la nouvelle table « query4 ».

Requête 4

Mapper

```
byte[] values = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("G"));
String val = Bytes.toString(values);
String[] key2 = Bytes.toString(key.get()).split("/");
int year = 9999 - Integer.valueOf(key2[0]);
String newkey = key2[2] + "/" + year;
this.text.set(val);
context.write(new Text(newkey), this.text);
```

Dans le mapper de cette requête, nous avons utilisé la table Grade, pour utiliser la clé avec l'année que nous avons inversée afin de l'utiliser dans le reducer. Nous avons changé la clé sous la forme codeMatiere/annee avec comme valeur la note.

Reducer

```

HTable courses = new HTable(conf, setting.getId()+"C");

int cumule = 0;
int total = 0;
int nb = 0;

for (Text vals : values) {
    if (Float.parseFloat(vals.toString()) >= 1000) {
        nb++;
        cumule += nb;
    }
    total++;
}

float valu = (float) Math.round(((float) nb / total)*100)/100;
String[] key2 = key.toString().split("/");
String matiere = key2[0];
int year = Integer.valueOf(key2[1]);
String key3 = matiere + "/" + (9999-year);

Put put = new Put(Bytes.toBytes(key3));
put.add(Bytes.toBytes("#"), Bytes.toBytes("N"), Bytes.toBytes(String.valueOf(foundname(matiere,year,courses,true))));
put.add(Bytes.toBytes("#"), Bytes.toBytes("NT"), Bytes.toBytes(String.valueOf(valu)));
put.add(Bytes.toBytes("#"), Bytes.toBytes("C"), Bytes.toBytes(String.valueOf(cumule)));
put.add(Bytes.toBytes("#"), Bytes.toBytes("T"), Bytes.toBytes(String.valueOf(total)));

courses.close();
context.write(null, put);

```

Ici, dans le reducer de la requête 4, nous calculons les taux de réussite selon la matière et l'année puis nous trouvons le nom de la matière avec la méthode `foundname(matiere,year,courses,true)`.

Methode foundname

```

Get get = new Get(Bytes.toBytes(key+"/"+year));
Result resultget = courses.get(get);
byte [] newnom = resultget.getValue(Bytes.toBytes("#"), Bytes.toBytes("N"));
String newn = Bytes.toString(newnom);
if(newn == null || newn.equals("") || newn.equals("null")) {
    if(year==7998) {
        flag=false;
        return foundname(key,year-1,courses,flag);
    }else {
        if(!flag) {
            return foundname(key,year-1,courses,flag);
        }else {
            return foundname(key,year+1,courses,flag);
        }
    }
}
return newn;

```

Cette méthode qui permet de trouver les noms des matières, elle recule dans le temps pour chercher une valeur à mettre dans le nom, par exemple si la matière se nomme « bigdata » en 2006, et en 2010, elle se nomme « cloud computing » et nous cherchons le nom pour l'année 2008, la méthode va faire en sorte que le nom de la matière soit « bigdata ». Si elle a une date antérieure n'a pas été trouvé (c'est-à-dire la matière est disponible mais n'a pas de nom dans les années antérieures), elle va chercher un nom de matière supérieur à la date la plus proche.

Requête 5

Mapper

```
byte[] values = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("G"));
String val = Bytes.toString(values);
String[] key2 = Bytes.toString(key.get()).split("/");
String programme = key2[1].substring(0, 2);
String prog = "";
if (programme.equals("01") || programme.equals("02")) {
    prog = "L1";
}

if (programme.equals("03") || programme.equals("04")) {
    prog = "L2";
}

if (programme.equals("05") || programme.equals("06")) {
    prog = "L3";
}

if (programme.equals("07") || programme.equals("08")) {
    prog = "M1";
}

if (programme.equals("09") || programme.equals("10")) {
    prog = "M2";
}

String newkey = key2[2] + "/" + key2[0] + "/" + prog;
this.text.set(val);
context.write(new Text(newkey), this.text);
```

Nous utilisons la table Grade pour répondre à la requête 5, nous utilisons le semestre afin de récupérer le programme, et nous créons une nouvelle clé qui est : codeMatiere/annee/programme avec la note en valeur

Reducer

```
HTable courses = new HTable(conf, setting.getId()+"C");
float total=0;
int nb = 0;
for (Text vals : values) {
    total += Float.parseFloat(vals.toString())/100;
    nb++;
}

String[] key2 = key.toString().split("/");
int year = 9999-Integer.parseInt(key2[1]);
String code = key2[0];

float valu = total/nb;
Put put = new Put (Bytes.toBytes(key.toString()));
put.add(Bytes.toBytes("#"),Bytes.toBytes("M"),Bytes.toBytes(String.valueOf(valu)));
put.add(Bytes.toBytes("#"),Bytes.toBytes("N"),Bytes.toBytes(String.valueOf(foundname(code,year,courses,true))));
context.write(null, put);
courses.close();
```

Dans ce reducer, nous calculons la moyenne pour chaque matière sur une année. Puis nous trouvons le nom de la matière et nous créons la nouvelle table, qui contient deux colonnes une avec le nom et l'autre avec la note avec la même clé.

Requête 6

Mapper

```
byte[] values = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("NT"));
byte[] names = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("N"));

String val = Bytes.toString(values);
String newkey = Bytes.toString(key.get());
text.set(val);

context.write((new Text(newkey+"/"+Bytes.toString(names))), text);
```

Pour la requête 6, nous utilisons la table de la requête 4, afin de former une nouvelle clé avec la key de la requête 4 et nous rajoutons le nom de la matière et comme valeur nous avons le taux de réussite.

Reducer

```
HTable courses = new HTable(conf, setting.getId()+"C");

String valfinal = "";
String[] key2 = key.toString().split("/");
String matiere = key2[0];
int year = Integer.valueOf(key2[1]);
String TEST = foundkey(matiere,9999-year,courses,true);

Get get = new Get(Bytes.toBytes(TEST));
Result result = courses.get(get);
String[] uouo = getColumnInColumnFamily(result, "I");
if (uouo != null) {
    for (String string : uouo) {
        byte[] newnom = result.getValue(Bytes.toBytes("I"), Bytes.toBytes(string));
        String newn = Bytes.toString(newnom);

        for (Text vals : values) {
            valfinal = vals.toString();
        }

        String key3 = matiere + "/" + year + "/" + newn;
        Put put = new Put(Bytes.toBytes(key3));
        put.add(Bytes.toBytes("#"), Bytes.toBytes("TR"), Bytes.toBytes(String.valueOf(valfinal)));
        put.add(Bytes.toBytes("#"), Bytes.toBytes("N"), Bytes.toBytes(String.valueOf(key2[2])));

        context.write(null, put);
    }
}
courses.close();
```

Nous récupérons la bonne clé avec la prise en compte des intervalle (c'est-à-dire si sur une année pour une matière, nous n'avons pas de valeur dans la table courses, nous remontons à une année où la valeur existe sinon on avance dans les années pour avoir une correspondance). Ensuite, nous allons chercher la liste qualifier de la famille « I » grâce à la méthode getColumnInColumnFamily afin de récupérer les noms des différents instructeurs et leur associer le taux de réussite de la matière qu'il enseigne. Enfin, nous mettons les valeurs dans la nouvelle table avec la clé matiere/annee/nomInstructor.

Méthode getColumnInColumnFamily

```
public String[] getColumnInColumnFamily(Result r, String ColumnFamily) {
    NavigableMap<byte[], byte[]> familyMap = r.getFamilyMap(Bytes.toBytes(ColumnFamily));
    if (familyMap == null) {
        return null;
    }
    String[] Quantifiers = new String[familyMap.size()];

    int counter = 0;
    for (byte[] bQuantifier : familyMap.keySet()) {
        Quantifiers[counter++] = Bytes.toString(bQuantifier);
    }

    return Quantifiers;
}
```

Cette méthode renvoie une liste de qualifiant d'une famille.

Requête 7

Mapper

```

byte[] values = value.getValue(Bytes.toBytes("#"), Bytes.toBytes("G"));
String val = Bytes.toString(values);
String[] key2 = Bytes.toString(key.get()).split("/");
String programme = key2[1].substring(0, 2);
String prog = "";
if (programme.equals("01") || programme.equals("02")) {
    prog = "L1";
}

if (programme.equals("03") || programme.equals("04")) {
    prog = "L2";
}

if (programme.equals("05") || programme.equals("06")) {
    prog = "L3";
}

if (programme.equals("07") || programme.equals("08")) {
    prog = "M1";
}

if (programme.equals("09") || programme.equals("10")) {
    prog = "M2";
}

String newkey = key2[1].substring(2) + "/" + key2[0] + "/" + prog;
this.text.set(val);
context.write(new Text(newkey), this.text);

```

Pour la dernière requête, dans ce code nous récupérons le programme, puis nous formons la nouvelle clé qui comprend : codeEtudiant/annee/programme avec comme valeur la note.

Reducer

```

int total = 0;
int nb = 0;

for (Text vals : values) {
    total+=Float.parseFloat(vals.toString());
    nb++;
}

float val1 = (float) total/nb;
int val=(int) (val1);

String note= String.format("%04d", val).substring(0, 2)+"."+String.format("%04d", val).substring(2);

String key3 = String.format("%04d", (2000-val))+"/"+key.toString();

Put put = new Put(Bytes.toBytes(key3));
put.add(Bytes.toBytes("#"),Bytes.toBytes("M"),Bytes.toBytes(note));

context.write(null, put);

```

Dans le reducer, nous utilisons la clé(row), que nous inversons ce qui nous permet de classer les élèves selon leurs notes et nous mettons la clé du map peut à la suite par exemple : un élève ayant 18.5 de moyenne aura une clé 0150/200900020/2010/L3 et comme valeur sa note 18.50.

Statistique de chaque Job

Job1

```
Map-Reduce Framework
  Map input records=1639842
  Map output records=1639842
  Map output bytes=57394470
  Map output materialized bytes=60674160
  Input split bytes=118
  Combine input records=0
  Combine output records=0
  Reduce input groups=42529
  Reduce shuffle bytes=60674160
  Reduce input records=1639842
  Reduce output records=42529
  Spilled Records=3279684
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=1980
  CPU time spent (ms)=132290
  Physical memory (bytes) snapshot=570892288
  Virtual memory (bytes) snapshot=4052688896
  Total committed heap usage (bytes)=303562752
```

Job2

```
Map-Reduce Framework
  Map input records=1639842
  Map output records=1639842
  Map output bytes=21317946
  Map output materialized bytes=24597636
  Input split bytes=118
  Combine input records=0
  Combine output records=0
  Reduce input groups=132
  Reduce shuffle bytes=24597636
  Reduce input records=1639842
  Reduce output records=132
  Spilled Records=3279684
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=356
  CPU time spent (ms)=11180
  Physical memory (bytes) snapshot=553283584
  Virtual memory (bytes) snapshot=4055846912
  Total committed heap usage (bytes)=321388544
```


Job3

```
Total negabyte milliseconds taken by all reduce tasks
Map-Reduce Framework
  Map input records=11903
  Map output records=11903
  Map output bytes=389357
  Map output materialized bytes=413169
  Input split bytes=122
  Combine input records=0
  Combine output records=0
  Reduce input groups=900
  Reduce shuffle bytes=413169
  Reduce input records=11903
  Reduce output records=900
  Spilled Records=23806
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=112
  CPU time spent (ms)=4690
  Physical memory (bytes) snapshot=549048320
  Virtual memory (bytes) snapshot=4021121024
  Total committed heap usage (bytes)=314572800
```

Job4

```
Map-Reduce Framework
  Map input records=1639842
  Map output records=1639842
  Map output bytes=29517156
  Map output materialized bytes=32796846
  Input split bytes=118
  Combine input records=0
  Combine output records=0
  Reduce input groups=11903
  Reduce shuffle bytes=32796846
  Reduce input records=1639842
  Reduce output records=11903
  Spilled Records=3279684
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=1487
  CPU time spent (ms)=83260
  Physical memory (bytes) snapshot=583385088
  Virtual memory (bytes) snapshot=4083548160
  Total committed heap usage (bytes)=342360064
```


Job5

```
Map-Reduce Framework
  Map input records=1639842
  Map output records=1639842
  Map output bytes=34436682
  Map output materialized bytes=37716372
  Input split bytes=118
  Combine input records=0
  Combine output records=0
  Reduce input groups=32732
  Reduce shuffle bytes=37716372
  Reduce input records=1639842
  Reduce output records=32732
  Spilled Records=3279684
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=3702
  CPU time spent (ms)=194260
  Physical memory (bytes) snapshot=582176768
  Virtual memory (bytes) snapshot=4078837760
  Total committed heap usage (bytes)=320864256
```

Job6

```
Map-Reduce Framework
  Map input records=11903
  Map output records=11903
  Map output bytes=429531
  Map output materialized bytes=453343
  Input split bytes=119
  Combine input records=0
  Combine output records=0
  Reduce input groups=11903
  Reduce shuffle bytes=453343
  Reduce input records=11903
  Reduce output records=23887
  Spilled Records=23806
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=1239
  CPU time spent (ms)=78200
  Physical memory (bytes) snapshot=592293888
  Virtual memory (bytes) snapshot=4064018432
  Total committed heap usage (bytes)=322437120
```

Job7

```
Map-Reduce Framework
  Map input records=1639842
  Map output records=1639842
  Map output bytes=39356208
  Map output materialized bytes=42635898
  Input split bytes=118
  Combine input records=0
  Combine output records=0
  Reduce input groups=42529
  Reduce shuffle bytes=42635898
  Reduce input records=1639842
  Reduce output records=42529
  Spilled Records=3279684
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=416
  CPU time spent (ms)=14440
  Physical memory (bytes) snapshot=613470208
  Virtual memory (bytes) snapshot=4076261376
  Total committed heap usage (bytes)=336592896
```