

HMIN313 : Introduction au système de gestion de données distribué

HBase (TP sur deux séances)

1. Enoncé

Nous abordons dans ce TP, quelques principes de base de mise en œuvre au sein du système HBase. HBase admet trois modes de déploiement : un mode autonome (ou local), un mode pseudo-distribué et enfin un mode pleinement distribué. Nous travaillerons dans un premier temps sur une instance HBase locale. Dans ce mode autonome, HBase exploite un système de fichier local et non pas HDFS. Les processus d'arrière plan (RegionServer, HMaster et ZooKeeper) sont regroupés et exécutés dans la même machine virtuelle Java JVM.

1. client JRuby : construction de tables et de familles de colonnes, et opérations insertion et suppression de données ("column qualifier", valeur, estampille)
2. client Java : construction de tables et de familles de colonnes, et opérations insertion et suppression de données ("column qualifier", valeur, estampille)

Vous exploiterez dans le cadre de ce td, à la fois le client JRuby et l'API Java. Pour ce faire, vous téléchargerez une version récente de HBase (voir [http ://hbase.apache.org/book/quickstart.html](http://hbase.apache.org/book/quickstart.html)). Vous démarrerez votre serveur HBase (exploitation en mode local).

2. Appropriation

Vous ferez en sorte de construire une table **Commune** caractérisée par deux familles de colonnes. Vous alimenterez cette table à partir de quelques tuples, en prenant la ville de Montpellier en exemple. Un exemple de table relationnelle Commune en extension vous est donnée en annexe. Vous traduirez cette table en table HBase.

2.1 Question 1

- opérer la construction à partir du client JRuby et explorer les commandes de définition et de manipulation des données (create, put, delete, list, scan, ...). L'aide en ligne est consultable avec les commandes `help 'ddl'` et `help 'dml'`, respectivement pour la création et la manipulation de données. Des exemples de manipulation d'une table 'personne' vous sont donnés en annexe. Vous reprendrez ces exemples dans le cadre de la définition de la table Commune.
- opérer le même travail à partir d'un projet Java sous Eclipse et des API dédiées. A cet effet, vous définirez une classe applicative dotée d'une méthode permettant d'afficher (scan) le contenu d'une colonne (dernière valeur) pour tous les tuples de la table Commune.

2.2 Question 2

Manipulez l'exemple de filtre fourni, et proposez un exemple de consultation qui pourrait avoir un sens dans le contexte de l'information sur les communes, par exemple retourner les communes qui ont plus de 150000 habitants en 2010 ou encore renvoyer le nombre de communes par numéro de département.

2.3 Question 3

Une fois les principes de construction et de consultation des tables acquis, réfléchissez à quelle conduite à tenir pour gérer des données enrichies par des descriptions provenant par exemple de l'Insee, du site des impôts ou bien de l'IGN

- proposer à ce titre la construction de nouvelles tables à partir du client JRuby ou Java

2.4 Question 4

Une fois, les modèles de données établis, élaborer une approche pour automatiser l'alimentation en données de votre système HBase. Vous pouvez par exemple poser votre réflexion, à partir des fichiers tabulés portant sur les communes.

3. Annexe

3.1 Extension Table Commune

CODE_I	NOM_COM	POP_1975	POP_2010
34248	SAINTE-CROIX-DE-QUINTILLARGUES	127.033553	561.720409
34249	SAINT-DREZERY	587.63955	2207.40611
34025	BASSAN	795.304324	1599.4016
34250	SAINT-ETIENNE-D'ALBAGNAN	253.31857	287.52352
34251	SAINT-ETIENNE-DE-GOURGAS	216.105575	489.314637
34252	SAINT-ETIENNE-ESTRECHOUX	406.570257	245.980104
34253	SAINT-FELIX-DE-L'HERAS	25.248617	36.8979349
34254	SAINT-FELIX-DE-LODEZ	519.740374	1209.95666
34255	SAINT-GELY-DU-FESC	2010.13406	8528.24947
34256	SAINT-GENIES-DES-MOURGUES	825.375932	1623.96003
34257	SAINT-GENIES-DE-VARENSAL	218.473421	209.311992
34167	MONTELS	127.655942	246.150356
34168	MONTESQUIEU	34.3551636	63.8836005
34169	MONTFERRIER-SUR-LEZ	1682.12888	3320.80271
34017	AUMES	307.185836	455.743575
34170	MONTOLUIERS	209.369513	237.112177
34171	MONTOLIEU	38.7548087	161.147619
34172	MONTPELLIER	191767.053	260959.188
34173	MONTPEYROUX	774.815254	1206.2612
34174	MOULES-ET-BAUCELS	131.447644	845.71166
34175	MOUREZE	78.5494361	176.923015
34176	MUDAISON	735.409812	2506.47942

Listing 1 – Un tuple

3.2 Exemple d'interaction avec l'interpréteur JRuby

```

Créer une table 'personne' avec une famille de colonnes 'general'.
Vérifier que la table a bien été créée puis afficher la description de la table
'personne'
Modifier la famille de colonnes 'general' pour qu'elle affiche jusqu'à 4 versions des
valeurs de ses colonnes.
Vérifier la modification en affichant à nouveau la description de la table.
Ajouter dans la table 'personne', le tuple '176073424356' avec la valeur 'Martin'
    pour la colonne 'general:nom'
Lister le contenu de la table 'personne'
Ajouter pour le tuple '176073424356', la valeur '39' pour la colonne 'general:age'
Ajouter dans la table 'personne', le tuple '2860738256345' avec la valeur 'Dupond'
    pour la colonne 'general:nom'
Ajouter une valeur quelconque pour la colonne 'general:prenom' du tuple
    '2860738256345', puis modifier la valeur de cette colonne 4 fois de suite.
Afficher le tuple '2860738256345'.
Afficher la valeur courante de la colonne '2860738256345'-'general:prenom'.
Afficher la valeur précédente de la colonne '2860738256345'-'general:prenom'.
Afficher toutes les valeurs précédentes de la colonne
    '2860738256345'-'general:prenom'.
Lister les lignes dont la clef commence par '286'.
Supprimer la cellule '176073424356'-'general:nom' et lister le contenu de la table.
Supprimer la famille de colonne 'general' de la table (à l'aide de la commande alter).
Supprimer la table 'personne'

--- réponses
create 'personne', 'general'
list
disable 'personne'
alter 'personne', NAME => 'general', VERSIONS => 4
enable 'personne'
describe 'personne'
put 'personne', '176073424356', 'general:nom', 'Martin'
scan 'personne'
put 'personne', '2860738256345', 'general:nom', 'Dupond'
put 'personne', '2860738256345', 'general:prenom', 'Philoe'
put 'personne', '2860738256345', 'general:prenom', 'Zoe'
put 'personne', '2860738256345', 'general:prenom', 'Arsinoe'
put 'personne', '2860738256345', 'general:prenom', 'Aglae'
put 'personne', '2860738256345', 'general:prenom', 'Berthe'
put 'personne', '2860738256345', 'general:prenom', 'Agathe'
get 'personne', '2860738256345'
get 'personne', '2860738256345', 'general:prenom'
get 'personne', '2860738256345', {COLUMN => 'general:prenom', TIMERANGE => [0,
    1442245190698]}
-- timestamp de Agathe : 1442245190698
get 'personne', '2860738256345', {COLUMN => 'general:prenom', TIMERANGE => [0,
    1442245190698], VERSIONS => 4}

scan 'personne', {FILTER => "PrefixFilter('286')"}
ou bien
scan 'personne', STARTROW => '286', ENDROW => '287'
delete 'personne', '176073424356', 'general:nom'
scan 'personne'
disable 'personne'
alter 'personne', 'delete' => 'general'
enable 'personne'
scan 'personne'

```

```
disable 'personne'  
drop 'personne'  
list
```

Listing 2 – Un tuple