

Apunts CE_5074 4.1

lloc: [Institut d'Ensenyaments a Distància de les Illes
Balears](#)

Curs: Sistemes de Big Data

Llibre: Apunts CE_5074 4.1

Imprès per: Carlos Sanchez Recio

Data: dimarts, 24 de desembre 2024, 15:08

Taula de continguts

1. Introducció

2. Bases de dades NoSQL i el model columnar

3. Què és HBase?

4. Arquitectura

4.1. Emmagatzematge de les regions

4.2. Catàleg de dades

5. Interactuar amb HBase

6. Llenguatge de Definició de Dades

7. Llenguatge de Manipulació de Dades

7.1. Filtres

7.2. Inserció de fitxers CSV

8. HBase i Hive

9. HBase i Impala

1. Introducció

En el mòdul de Sistemes de Big Data vàrem introduir les bases de dades NoSQL. La principal característica és que permeten treballar amb dades d'una manera molt flexible, sense un esquema fixat. Normalment les bases de dades NoSQL es fan servir en entorns amb volums molt grans de dades, amb requeriments d'alt rendiment i d'alta disponibilitat.

En aquest lliurament ens centrarem en HBase, la base de dades NoSQL que s'executa sobre HDFS, dins de l'ecosistema Hadoop. Aquesta és una base de dades que segueix el model columnar (o *wide-column*).

Veurem l'arquitectura d'HBase, com interactuar amb el seu *shell* i amb la interfície gràfica de Hue, i com crear taules i manipular les seves dades. Finalment explicarem també com podem treballar de forma conjunta amb Hive i HBase, de manera que les dades del nostre magatzem de dades estiguin emmagatzemades en HBase, però amb la possibilitat d'interactuar de manera més senzilla amb les dades mitjançant HiveQL.

2. Bases de dades NoSQL i el model columnar

En el primer lliurament del mòdul de Sistemes de Big Data vàrem veure què són les bases de dades NoSQL i vàrem introduir els diferents models de dades que aquestes implementen. En aquell moment vàrem fer més èmfasi en el model basat en documents, ja que és el que fa servir MongoDB. Després, en el lliurament 3, ens vàrem centrar en el model basat en grafs.

En canvi, ara veurem amb una mica més de detall el model columnar o *wide-column* que és el que utilitza HBase. En aquest model es fan servir taules, files i columnes, però a diferència de les bases de dades relacionals, cada fila pot tenir diferents columnes (diferent número, noms i tipus). Segueix, per tant, un enfocament *schemaless* (sense un esquema fixat), la qual cosa ens dona molta flexibilitat.

En la següent imatge podem veure un exemple d'una taula amb el model columnar. Conté tres files. La primera té tres columnes A, B i C, cada una amb el seu valor; la segona fila té les columnes B, D i E; mentre que la tercera fila té dues columnes, A i F. És a dir, cada fila pot tenir una estructura diferent a la resta.

Fila 1	Columna A	Columna B	Columna C
	valor	valor	valor
Fila 2	Columna B	Columna D	Columna E
	valor	valor	valor
Fila 3	Columna A	Columna F	
	valor	valor	

Imatge: Taula amb un model columnar

Com que cada fila pot tenir diferents columnes, hem de saber cada valor a quina columna fa referència. Per tant, cada cella de la taula no conté només un valor (com en una taula d'una base de dades relacional), sinó que també ha de contenir el nom de la seva columna. És a dir, una cella serà una parella *<nom de columna, valor>*.

Tot i que cada fila pot tenir diferents columnes, les bases de dades columnars (i HBase en particular) permeten definir el que s'anomenen famílies de columnes. Són agrupacions de columnes que tenen una temàtica comú. Suposem que tenim una taula client. Ens pot interessar tenir una família de columnes relacionada amb les seves dades d'identitat (on tendrien columnes com el nom, llinatges, NIF, etc.), una altra amb les seves dades de contacte (amb columnes com el telèfon, el correu electrònic, l'adreça postal) i una altra amb les dades de pagament (per exemple, amb una columna per al número de targeta de crèdit).

Això no vol dir que totes les files tinguin columnes de totes les famílies. Ni tampoc que totes les files tinguin totes les columnes d'una família. Però sí que ens ajuda a establir una mínima estructura lògica. Podem veure un exemple en la taula següent:

	Família dades identitat			Família dades contacte			Família dades pagament
	Nom	Llinatges	NIF	Telèfon	Mail	Adreça	Targeta
Fila 1	Pep	Pérez	12345678A	611111111	pepperez@gmail.com	C/ ...	4299 ...
Fila 2	Aina	Álvarez		ainaal@gmail.com	Avda. ...		
Fila 3	Toni	Tomàs	23456789B	622222222			

Imatge: Taula amb famílies de columnes

Per altra banda, tot i que no ho hem representat en les taules anteriors, totes les files d'una taula han de tenir una columna que serveix com a clau. Una representació més fidel seria aquesta:

Clau	Família dades identitat			Família dades contacte			Família dades pagament
Clau	Nom	Llinatges	NIF	Telèfon	Mail	Adreça	Targeta
1	Pep	Pérez	12345678A	611111111	pepperez@gmail.com	C/ ...	4299 ...
Clau	Nom	Llinatges		Mail	Adreça		
2	Aina	Álvarez		ainaal@gmail.com	Avda. ...		
Clau	Nom	Llinatges	NIF	Telèfon			
3	Toni	Tomàs	23456789B	622222222			

Imatge: Taula amb claus i famílies de columnes

A més d'HBase, altres bases de dades columnars molt conegudes i utilitzades són BigTable de Google i Apache Cassandra. De fet, BigTable és la precursora de totes les altres bases de dades d'aquest model, que estan en gran mesura basades en ella. Tan és així que alguns autors xerren de bases de dades de tipus BigTable per referir-se al conjunt del que aquí hem denominat bases de dades columnars.

3. Què és HBase?

Apache HBase és una base de dades NoSQL de codi obert, que es basa en el model de dades columnar. Escrita en Java, és una base de dades distribuïda i escalable, dissenyada per emmagatzemar grans quantitats de dades en clústers de servidors. La seva arquitectura està basada en el sistema de fitxers distribuït HDFS.

La primera versió d'HBase es va llançar el 2008, molt basada en BigTable, la base de dades columnar que Google utilitza amplament en els seus productes. A partir del 2010 va passar a ser un subprojecte del projecte Hadoop, de l'Apache Software Foundation.

Tot i que HDFS permet emmagatzemar, processar i gestionar grans volums de dades d'una manera distribuïda i eficient, només permet un processament per lots (*batch*), amb un accés seqüencial a les dades. És per això que sorgeix la necessitat de comptar amb una eina com HBase, que proporcioni un accés aleatori (no seqüencial) i en temps real a les dades.

Resumint, els grans avantatges que ofereix HBase són:

- Permet treballar amb datasets molt grans (fins a petabytes)
- És escalable pel que fa als recursos de maquinari
- És molt efectiu pel que fa al cost per a grans volums de dades
- Té una gran consistència: no suporta completament totes les característiques ACID (*Atomicity, Consistency, Isolation* i *Durability*), però sí proporciona lectures i escriptures consistents
- És una solució per a processament en temps real
- És una solució d'alta disponibilitat
- És una solució distribuïda que executa les queries de manera paral·lela en els clústers

HBase és amplament utilitzada en l'àmbit del processament de dades massives, especialment en temps real. També és molt popular en el sector de la web, les xarxes socials i el comerç electrònic.

Moltes de les grans companyies utilitzen HBase. Facebook emmagatzema tots els missatges del seu servei de missatgeria en HBase; Twitter també s'executa sobre HBase en un clúster Hadoop; i Mozilla el fa servir per gestionar totes les dades de fallades del navegador. Són només alguns exemples.



Podeu consultar la documentació oficial d'HBase a <https://hbase.apache.org/book.html>

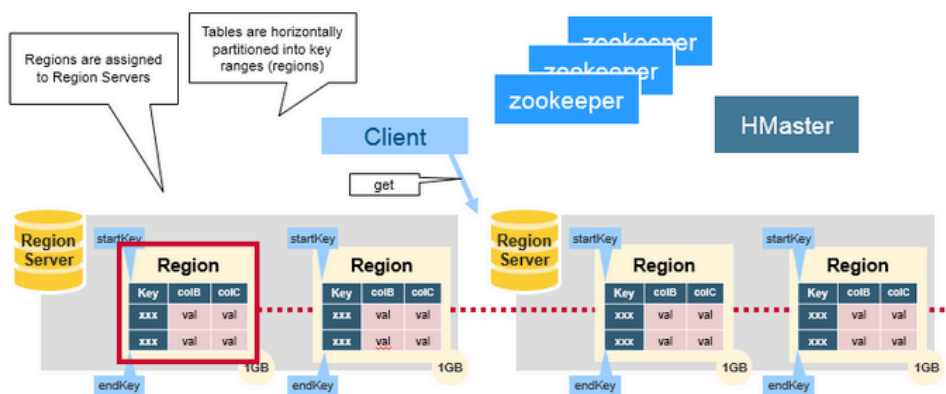
4. Arquitectura

En HBase les taules es particionen horitzontalment en el que anomenam **regions**, que són els blocs fonamentals d'un clúster HBase. Una regió és un conjunt ordenat de files d'una taula, que queden delimitades per una clau inicial i una clau final.

Inicialment, una taula ocupa una regió i a mesura que la taula creix i té més files, quan s'arriba a la mida màxima permesa, la regió es divideix (*region splitting*) en dues a partir de la clau que queda just al mig, de manera que les dues regions siguin igual de grans. En les versions actuals d'HBase, la mida màxima d'una regió per defecte és d'1 GB, tot i que es pot modificar canviant la propietat *hbase.hregion.max.filesize*. La documentació d'HBase recomana per treballar amb taules molt grans especificar una mida màxima d'entre 10 i 20 GB, essent l'òptim entre 5 i 10.

Cada regió d'una taula s'assigna a un dels nodes del clúster HBase. Aquests nodes són els anomenats **servidors de regions (Region Servers)**, o HRegionServers. Així doncs, un servidor de regions és l'encarregat de gestionar un conjunt de regions i emmagatzemar-les en HDFS (veurem en detall com ho fan en l'apartat 4.1). El servidor de regions es qui rep i serveix les peticions dels clients de lectura i escriptura sobre les seves regions.

Una taula pot tenir regions repartides en diferents servidors de regions. Però cada regió està servida per un únic servidor de regions: en HBase no hi ha replicació de regions. Però com que les regions estan emmagatzemades en HDFS, els blocs HDFS que formen cada regió sí que estaran replicats en múltiples DataNodes del clúster. És a dir, la replicació per tal d'oferir una alta disponibilitat la tenim a nivell d'HDFS, no d'HBase.

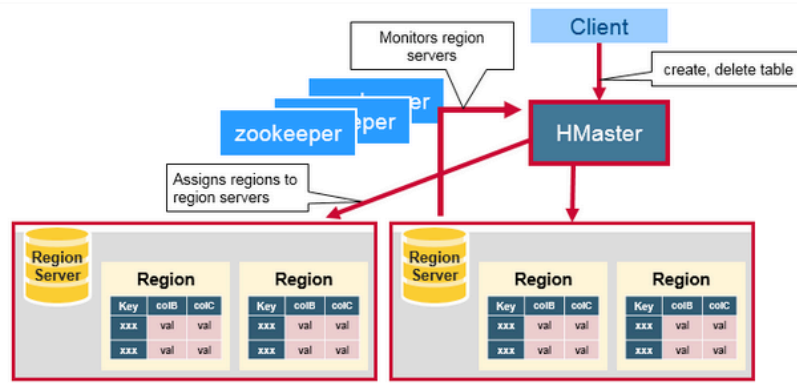


Imatge: Servidors de regions en un clúster HBase. Font: GitBook <https://nag-9-s.gitbook.io/hbase>

L'encarregat d'assignar una regió a un dels servidors de regions determinat és el **servidor mestre** del clúster (**Master Server**), o HMaster. En un clúster hi ha molts servidors de regions però només un únic servidor mestre actiu (tot i que es poden configurar servidors mestres passius per si cau el primari). Les seves tasques són:

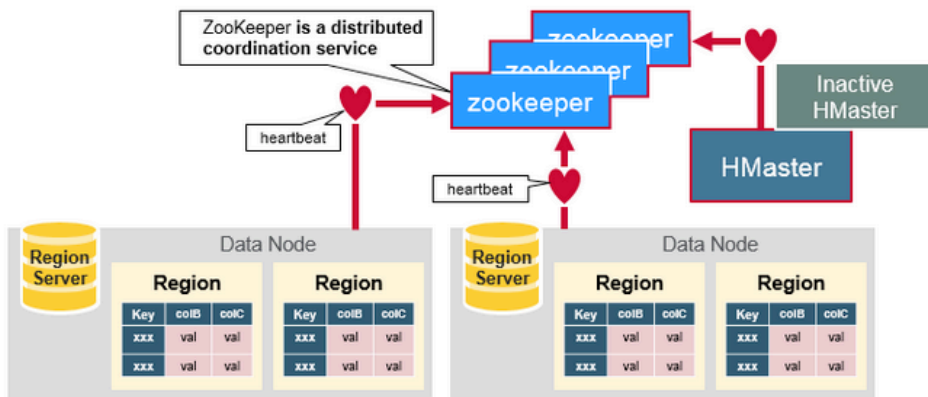
- Assigna regions als servidors de regions
- Du a terme un balaceig de regions entre els servidors de regions: descarrega servidors de regions molt ocupats i passa regions als menys ocupats
- Manté un estat del clúster
- És el responsable de les operacions que suposin canvis en les metadades com ara la creació d'una taula o d'una família de columnes

Normalment el servidor mestre s'executa sobre el NameNode i els servidors de regions sobre els DataNodes del clúster Hadoop.



Imatge: Servidor mestre en un clúster HBase. Font: GitBook <https://nag-9-s.gitbook.io/hbase>

A més dels servidors de regions i del servidor mestre, l'arquitectura d'HBase té un tercer component principal. Es tracta de **ZooKeeper**, un projecte d'Apache que ofereix un servei per a la coordinació de processos distribuïts. Zookeeper rep els *heartbeats* que li envien els servidors de regions (i també el mestre) per tal de saber quins servidors estan actius i disponibles i de llençar notifikacions en cas de caigudes de servidors. Zookeeper utilitza el consens per garantir l'estat dels nodes del clúster. És per això que hi ha múltiples nodes ZooKeeper en el clúster (normalment 3 o 5). Tota aquesta informació que proporcionen els nodes ZooKeeper les fa servir el servidor mestre per descobrir quins són els servidors de regions actius per tal d'enviar-los les regions.



Imatge: Nodes ZooKeeper en un clúster HBase. Font: GitBook <https://nag-9-s.gitbook.io/hbase>

Per altra banda, els clients es comuniquen amb els servidors de regions també fent servir ZooKeeper. En resum, ZooKeeper juga un paper clau com a coordinador del clúster.

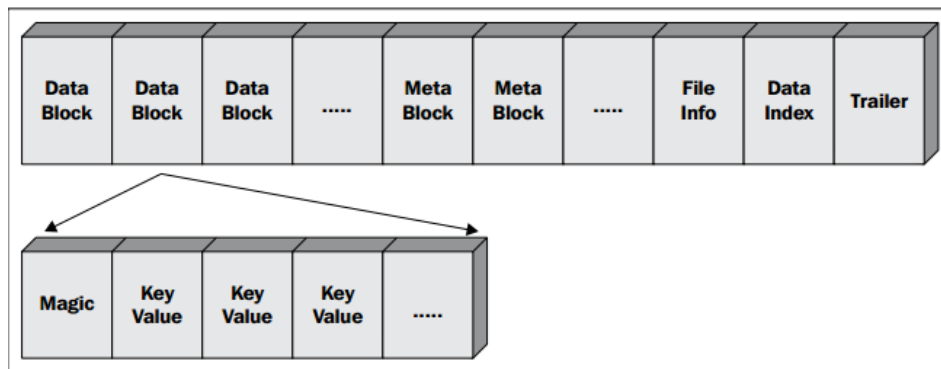
4.1. Emmagatzematge de les regions

Anteriorment hem vist que cada servidor de regions és l'encarregat d'emmagatzemar les seves dades en HDFS.

Cada servidor de regions té dos components compartits entre totes les regions: el **HLog** i el **BlockCache**. El HLog, també anomenat **WAL** (Write-Ahead Log) és el que dona a HBase la durabilitat de les dades en cas de fallada.

Cada escriptura en HBase s'emmagatzema primer en el HLog i queda allà fins que és escrita en disc. Aquest HLog està dins HDFS i per tant, està replicat a nivell de blocs HDFS. Per altra banda, el BlockCache és un espai de memòria on HBase manté una caché dels blocs de disc llegits més freqüentment.

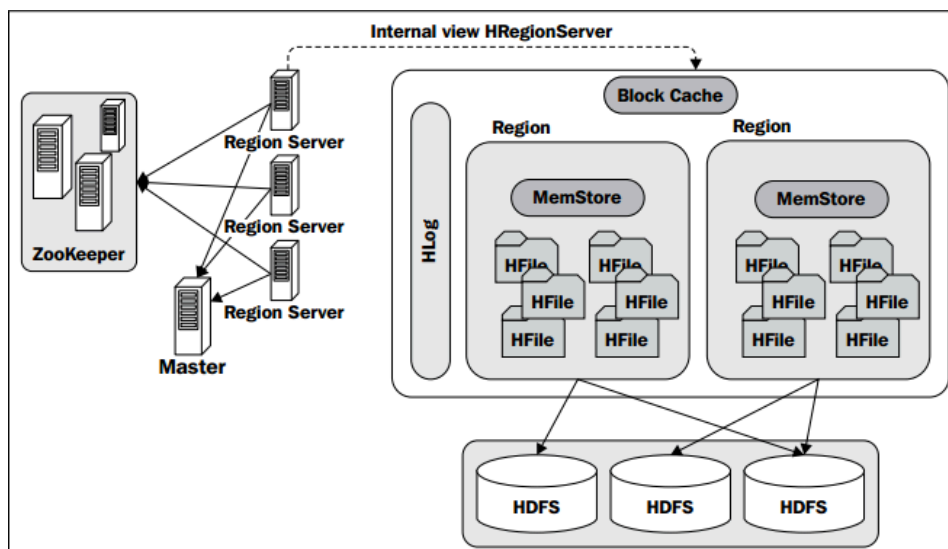
A més, cada regió es particiona verticalment segons les seves famílies de columnes del model lògic. Cada una d'aquestes famílies de columnes dóna lloc al que s'anomena magatzem o **store** (o HStore). D'aquesta manera els stores proporcionen un nivell d'aïllament físic per a les famílies de columnes. Cada un d'aquests stores conté múltiples fitxers, anomenats **HFiles** o **StoreFiles**. A la vegada, aquests HFiles es divideixen internament en un número variable de blocs de dades més petits, de 64 KB, que contenen una capçalera (*magic*) i un seguit de parelles clau-valor serialitzades. Després dels blocs de dades, els HFiles també tenen un número fix de blocs de metadades, d'índex i una marca de final.



Imatge: Divisió d'un HFile en blocs. Font: GitBook <https://nag-9-s.gitbook.io/hbase>

Finalment, tots aquests blocs són els que es van escrivint en HDFS. Recordem que en HDFS les dades es fragmenten en blocs HDFS, que per defecte tenen una mida de 128 MB. Per tant, en un bloc HDFS hi caben fins a 2.048 blocs d'HFiles. Recordem també que cada bloc HDFS es replica per defecte en 3 DataNodes, amb la qual cosa, els blocs dels HFiles acaben replicats al llarg del clúster HBase.

A més de tots els HFiles, el servidor de regions també guarda una caché d'escriptura per a cada regió, anomenada **MemStore**. Aquesta caché emmagatzema totes les dades que encara no s'han escrit en el disc.

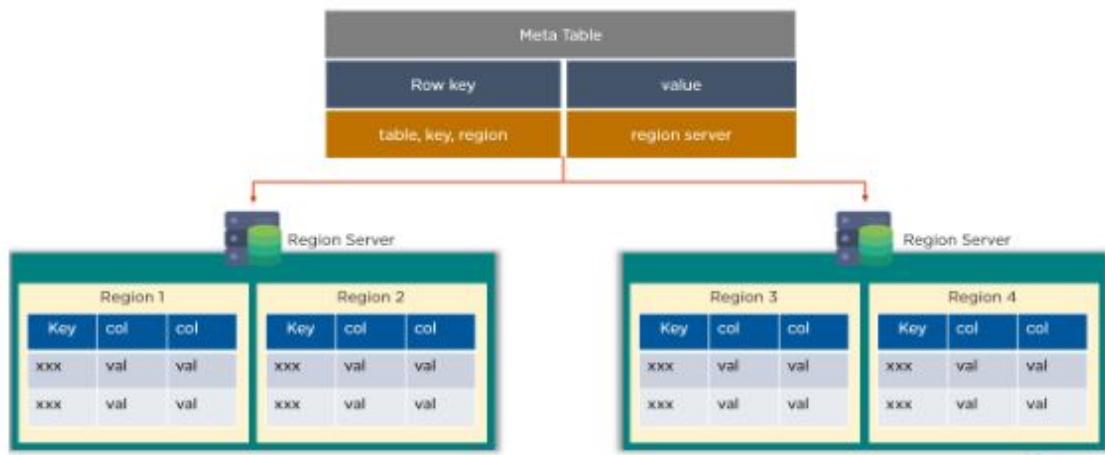


Imatge: *Visió interna d'un Region Server en un clúster HBase*

4.2. Catàleg de dades

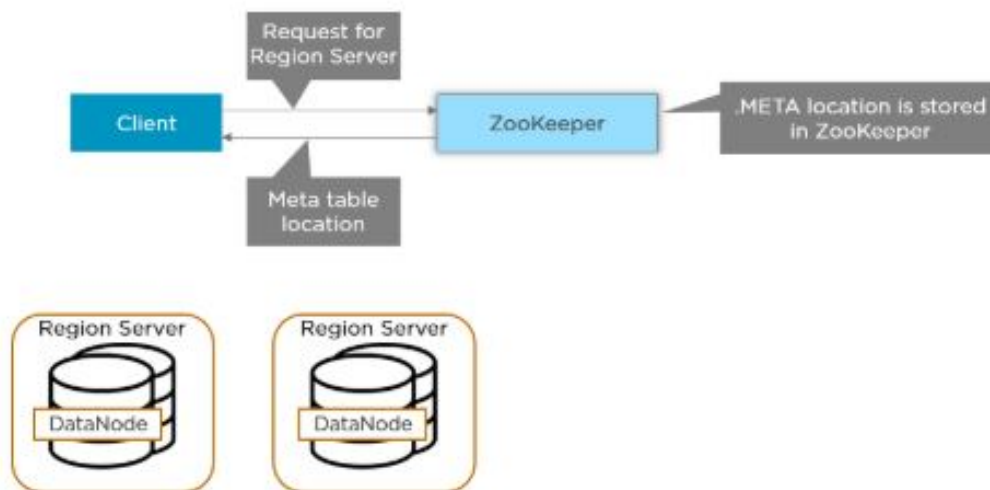
HBase manté una taula especial que serveix de catàleg. És tracta de la taula *hbase:meta* (es diu *meta* i està en l'espai de noms *hbase*), que guarda la informació de quins servidors de regions gestionen cada una de les regions de totes les taules que HBase gestiona.

Aquesta taula consisteix en una llista d'entrades de tipus clau-valor, on la clau té la forma *<taula, clau inicial de regió, id de regió>*, mentre que el valor és la ruta del servidor de regions corresponent. Els clients utilitzen aquesta taula per, donada una fila d'una taula, primer saber quina clau li correspon, i a partir d'ella saber quin és el servidor de regions on està guardada.



Imatge: Taula meta. Font: <https://www.simplilearn.com/tutorials/hadoop-tutorial/hbase>

Meta és una taula i com a tal també està emmagatzemada en un servidor de regions. Així que primer de tot, abans de poder cercar dins la taula meta, els clients necessiten saber a quin servidor de regions es troba. És ZooKeeper qui guarda aquesta informació.



Imatge: Determinant la ubicació de la taula meta

5. Interactuar amb HBase

La nostra màquina virtual de Cloudera QuickStart amb la que hem estat treballant durant el curs també té HBase instal·lat. Lògicament aquí tenim un clúster HBase d'un únic node, que té una instància dels tres serveis dels que hem parlat anteriorment: servidor mestre, servidor de regions i ZooKeeper, totes executant-se sobre una mateixa màquina virtual Java.

Tenim diverses maneres d'interactuar amb HBase en la nostra màquina virtual:

- Mitjançant el **HBase Shell** des d'una consola
- Des de **Hue**, amb una interfície gràfica
- Amb l'**API nativa de Java**, de la qual no parlarem en aquest curs
- Amb altres connectors per a altres llenguatges. Un d'ells és **HappyBase** per a Python, que es proposa per a l'activitat d'ampliació.

HBase Shell

Per entrar en el HBase Shell simplement hem d'escriure l'ordre següent en una consola:

```
hbase shell
```

```
[cloudera@quickstart ~]$ hbase shell
2023-03-22 05:29:17,721 INFO [main] Configuration.deprecation: hadoop.native.lib
is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017

hbase(main):001:0> █
```

Imatge: HBase Shell

Aquí podem escriure qualsevol de les ordres que permeten interactuar amb HBase. En els següents apartats anirem veient les principals ordres per crear, modificar i esborrar taules (llenguatge de definició de dades) i per a consultar i modificar les dades (llenguatge de manipulació de dades). A més, tenim algunes ordres de propòsit general:

- **status**: dona el status actual d'HBase, per exemple, el número de servidors
- **version**: dona la versió d'HBase que s'està utilitzant
- **help**: dona ajuda sobre les ordres d'HBase Shell
- **whoami**: dona informació sobre l'usuari actual
- **exit**: tanca la sessió de shell

```
hbase(main):001:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load

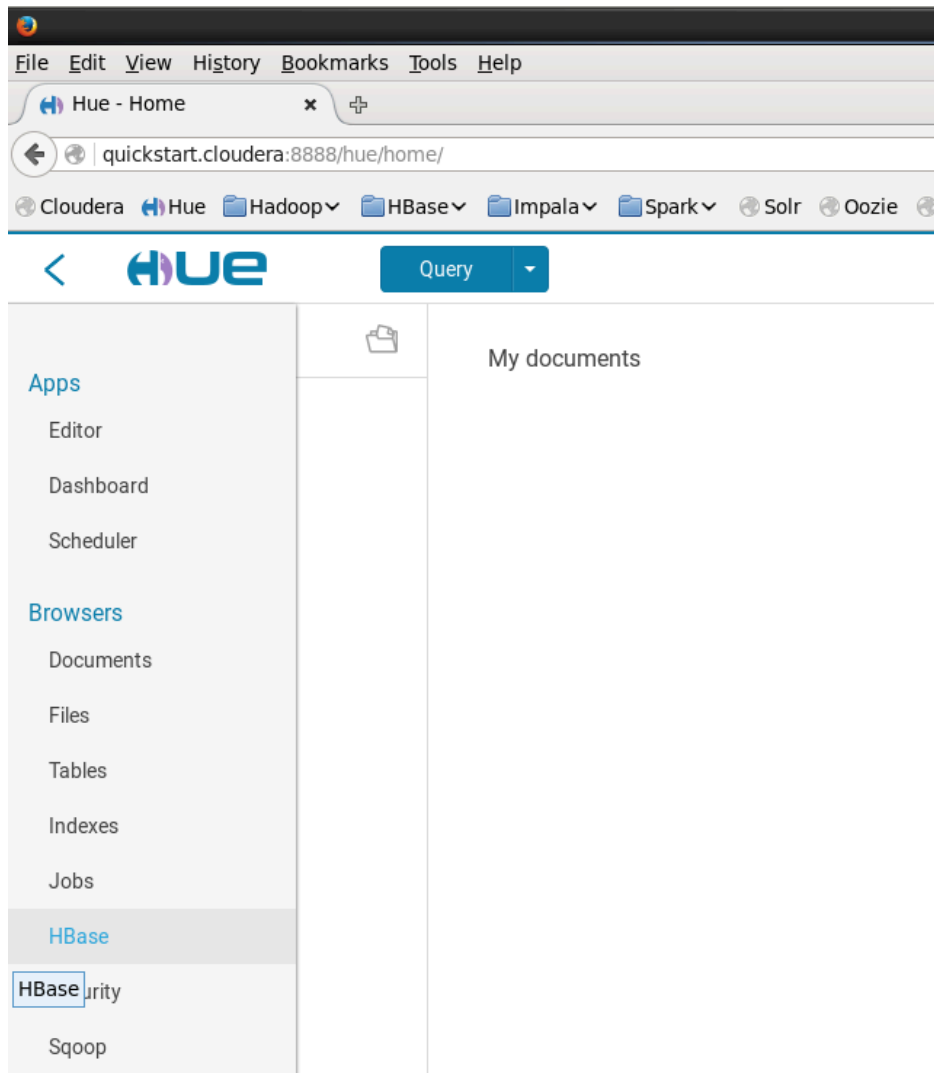
hbase(main):002:0> version
1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017

hbase(main):003:0> whoami
cloudera (auth:SIMPLE)
groups: cloudera, default
```


Imatge: Execució d'algunes ordres de propòsit general

Interfície gràfica de Hue

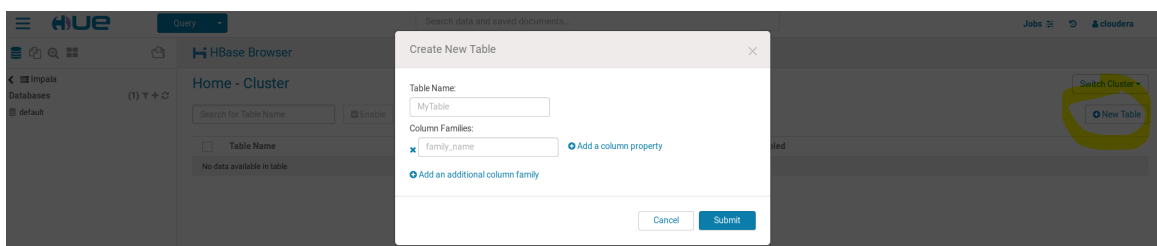
Per entrar en la interfície de Hue per a HBase, hem de desplegar el menú de les tres línies horitzontals que trobam a la part superior esquerra de Hue.



Imatge: Obrir la interfície de Hue per a HBase

També podem entrar en la interfície Hue per a HBase fent clic al botó .

Des d'aquí podem veure les taules definides en HBase i podem també crear-ne una de nova amb el botó New Table.



Imatge: Crear nova taula en la interfície de Hue

En la imatge següent podem veure com es veu una taula (*empleat*) que té 3 files amb dues famílies de columnes (*dades_personals* i *dades_professionals*) i cada una d'elles amb dues columnes. Veurem els detalls d'aquesta taula en els apartats 6 i 7, ja que l'emprarem per fer alguns exemples.

The screenshot shows the Hue HBase Browser interface. The top navigation bar includes the Hue logo, a 'Query' dropdown, and a search bar. The left sidebar shows 'Tables' with a message 'The database has no tables'. The main content area is titled 'Home - Cluster / empleat'. A query is entered in the search bar: 'row_key, row_prefix = escan, len [part, family:col2, len3, col_prefix = >3, len: col2 to col3] (Filter 1) AND Filter(2)'. The results are displayed in a table with 10 rows. The first three rows are visible, showing columns: 'empleat_professional', 'salari', 'empleat_persona', 'nom', 'empleat_professional', 'departamen', 'empleat_persona', 'ciutat'. The data for the first three rows is: 1. 35888, Joan, Sistemes, Elvissa; 2. 48888, Aina, Desenvolupament, Inca; 3. 38888, Pep, Desenvolupament, Rac. At the bottom, it says 'Fetched 10 entries starting from null in 0.142 seconds.' and there are buttons for 'Drop Rows', 'Bulk Upload', and 'New Row'.

empleat_professional	salari	empleat_persona	nom	empleat_professional	departamen	empleat_persona	ciutat
35888	Joan	Sistemes	Elvissa				
48888	Aina	Desenvolupament	Inca				
38888	Pep	Desenvolupament	Rac				

Imatge: Vista d'una taula HBase en Hue

Podem veure que la interfície també permet esborrar files i afegir-ne de noves, així com fer un upload d'un arxiu (veurem com en l'apartat 7.2).

Interfície web del servidors mestre i de regions

Per altra banda, també podem entrar a veure la interfície web tant del servidor mestre com del servidor de regions. Tenim dos marcadors al Firefox de la màquina virtual.

A <http://quickstart.cloudera:60010/master-status> tenim la interfície del Master Server, on entre d'altres, podem veure els servidors de regions registrats (només n'hi ha un). Si n'hi haguessin, aquí podríem veure els servidors mestre de backup, per al cas que falli el principal. També podem veure les taules creades en HBase (en aquest cas només una petita taula anomenada *empleats*), així com les tasques actives.

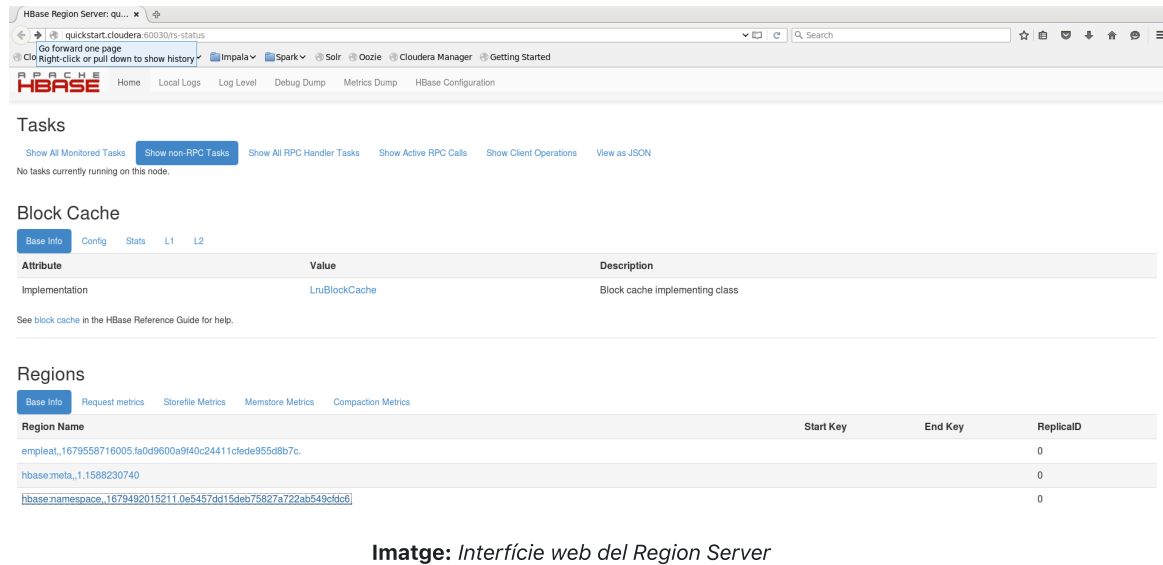
The screenshot shows the HBase Master Server web interface. The top navigation bar includes the HBase logo and links to Home, Table Details, Procedures, Local Logs, Log Level, Debug Dump, Metrics Dump, and HBase Configuration. The main content area is titled 'Master quickstart.cloudera'. Below this, there is a 'Region Servers' section with tabs for 'Base Stats', 'Memory', 'Requests', 'Storefiles', and 'Compactions'. The 'Base Stats' tab is selected, showing a table with columns: 'ServerName', 'Start time', 'Version', 'Requests Per Second', and 'Num. Regions'. The data for the first row is: quickstart.cloudera.60020,1679487868020, Wed Mar 22 05:24:28 PDT 2023, 1.2.0-cdh5.13.0, 0, 0. Below this, there is a 'Backup Masters' section with a table showing 'ServerName', 'Port', and 'Start Time'. The data for the first row is: Total:0. At the bottom, there is a 'Tables' section with tabs for 'User Tables', 'System Tables', and 'Snapshots'.

ServerName	Start time	Version	Requests Per Second	Num. Regions
quickstart.cloudera.60020,1679487868020	Wed Mar 22 05:24:28 PDT 2023	1.2.0-cdh5.13.0	0	0
Total:1			0	0

ServerName	Port	Start Time
Total:0		

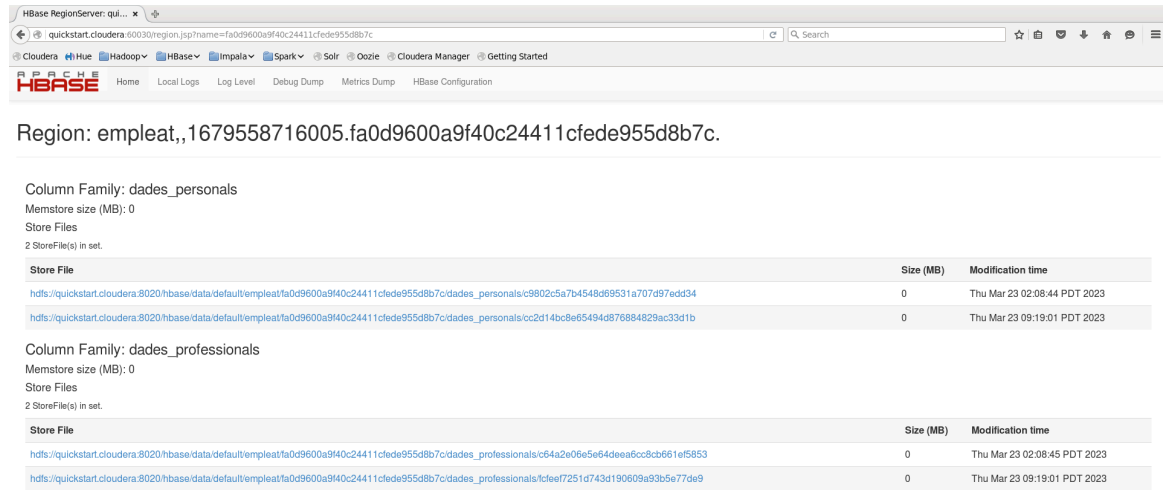
Imatge: Interfície web del Master Server

A <http://quickstart.cloudera:60030/rs-status> podem veure la interfície del Region Server. Aquí es mostren, entre d'altres les regions que està servint: En la següent imatge es veuen tres: una corresponent a una taula de dades anomenada *empleat*, una de la taula *hbase:meta* (catàleg de dades) i una altra amb la informació dels espais de noms, *hbase:namespace*.



Imatge: Interfície web del Region Server

Fent clic sobre cada una d'aquestes regions, podem veure els seus detalls. En la següent imatge es mostra la informació de la regió de la taula empleat. Podem observar que està organitzada en dues famílies de columnes i que, per a cada família té dos HFiles.



Imatge: Interfície web del Region Server. Detalls d'una regió

6. Llenguatge de Definició de Dades

Create

El primer que farem és crear una nova taula. Per això emprarem l'ordre **create** i haurem d'especificar el nom de la taula i totes les famílies de columnes que vulguem.

Anam a treballar amb una taula empleat, que tindrà dues famílies de columnes: dades personals i dades professionals. L'ordre corresponent serà:

```
create 'empleat', 'dades_personals', 'dades_professionals'
```

```
hbase(main):004:0> create 'empleat', 'dades_personals', 'dades_professionals'
0 row(s) in 1.3320 seconds
=> Hbase::Table - empleat
```

Imatge: Ordre create

Create_namespace

En HBase no hi ha els conceptes de base de dades ni d'esquema, habituals en tots els sistemes gestors de bases de dades relacionals, que ens permeten aïllar un conjunt de taules, típicament que pertanyen a una mateixa aplicació. En HBase totes les taules, en principi, estan al mateix nivell. En canvi, el que sí que podem és definir espais de noms, per separar les taules d'una aplicació de la resta. Podem crear un nou espai de noms amb l'ordre **create_namespace**:

```
create_namespace 'aplicacio1'
```

I després podríem definir taules emprant aquest espai de noms. Per exemple:

```
create 'aplicacio1:empleat', 'dades_personals', 'dades_professionals'
```

List

Podem veure el llistat de totes les taules que tenim en HBase amb l'ordre **list**:

```
hbase(main):006:0* list
TABLE
empleat
1 row(s) in 0.0150 seconds
=> ["empleat"]
```

Imatge: Ordre list

Exists

També podem comprovar si una taula existeix mitjançant l'ordre **exists**:

```
hbase(main):009:0> exists 'empleat'
Table empleat does exist

0 row(s) in 0.0210 seconds
```

Imatge: Ordre exists

Describe

I amb l'ordre **describe** podem veure la descripció d'una taula:


```
hbase(main):007:0> describe 'empleat'
Table empleat is ENABLED
empleat
COLUMN FAMILIES DESCRIPTION
{NAME => 'dades_personals', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW',
  REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS
=> '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', I
N_MEMORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'dades_professionals', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'R
OW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSI
ONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536
', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
2 row(s) in 0.1350 seconds
```

Imatge: Ordre describe

Disable, enable, is_disabled i is_enabled

Si volem esborrar o modificar les propietats d'una taula, abans hem de desactivar-la mitjançant l'ordre **disable**. Quan una taula està desactivada, les ordres *list* i *exists* ens diran que continua existint, però no podrem llegir les seves files (amb *scan*). Podem comprovar si una taula està desactivada amb l'ordre **is_disabled**. Una vegada desactivada, podem tornar-la a activar amb l'ordre **enable**. I podem comprovar que està activada amb **is_enabled**.

```
hbase(main):013:0> disable 'empleat'
0 row(s) in 2.2800 seconds

hbase(main):014:0> is_disabled 'empleat'
true
0 row(s) in 0.0200 seconds

hbase(main):015:0> enable 'empleat'
0 row(s) in 1.2890 seconds

hbase(main):016:0> is_enabled 'empleat'
true
0 row(s) in 0.0140 seconds
```

Imatge: Ordres per desactivar i activar una taula

Disable_all i enable_all

Podem desactivar (o activar) varies taules a la vegada fent servir l'ordre **disable_all** (o **enable_all**) i una expressió regular. Per exemple, totes les que comencin per 'e':

```
hbase(main):019:0> disable_all 'e.*'
empleat

Disable the above 1 tables (y/n)?
y
1 tables successfully disabled

hbase(main):020:0> enable_all 'e.*'
empleat

Enable the above 1 tables (y/n)?
y
1 tables successfully enabled
```

Imatge: Ordres disable_all i enable_all

Alter

Una vegada tenim una taula creada, podem modificar les seves propietats amb l'ordre **alter**. Primer, però, hauríem de desactivar la taula. Vegem, per exemple, com afegir una nova família de columnes 'dades_contacte':

```
alter 'empleat', 'dades_contacte'
```

Podem modificar alguna de les propietats d'aquesta nova família. Per exemple, modificarem VERSIONS, que indica el número màxim de versions que guardam del valor d'una cel·la (per defecte és 1):

```
alter 'empleat', NAME => 'dades_contacte', VERSIONS => 5
```

Finalment, esborrarem la família de columnes:

```
alter 'empleat', NAME => 'dades_contacte', METHOD => 'delete'
```

Drop i drop_all

Per acabar, si volem esborrar una taula, emprarem l'ordre **drop**. Recordau que la taula ha d'estar desactivada.

```
hbase(main):007:0> drop 'empleat'  
0 row(s) in 1.3290 seconds
```

Imatge: *Ordre drop*

Amb **drop_all** podem esborrar totes les taules que compleixin una expressió regular. Per exemple, per esborrar totes les taules que comencin per 'e':

```
drop_all 'e.*'
```

7. Llenguatge de Manipulació de Dades

Inserir i modificar dades: put

Tornem a crear la taula 'empleat' que hem emprat en l'apartat anterior. I ara volem inserir les dades que es mostren en aquesta taula:

Clau de fila	Família de columnes <i>dades_personals</i>		Família de columnes <i>dades_professionals</i>	
id_empleat	nom	ciutat	departament	salari
1	Joan	Eivissa	Sistemes	35000
2	Aina	Palma	Desenvolupament	40000
3	Pep	Maó	Desenvolupament	30000

Per fer-ho emprarem l'ordre **put**, que permet inserir el valor d'una cel·la d'una taula. Té la següent sintaxi:

```
put 'nom_taula', 'clau de fila', 'familia:columna', 'valor'
```

Vegem com inserir la primera fila, mitjançant les següents 4 ordres, una per a cada cel·la:

```
put 'empleat', '1', 'dades_personals:nom', 'Joan'
put 'empleat', '1', 'dades_personals:ciutat', 'Eivissa'
put 'empleat', '1', 'dades_professionals:departament', 'Sistemes'
put 'empleat', '1', 'dades_professionals:salari', '35000'
```



És important entendre que el que guarda HBase són cel·les, és a dir parelles clau-valor. Per tant, una inserció no és escriure una fila sencera, sinó una de les cel·les. I per tant, per inserir tota una fila, hem d'executar múltiples ordres *put*, una per a cada cel·la.

També és important recordar que no sempre totes les files tendran les mateixes columnes. Potser ens trobam que en una altra fila l'empleat no té la columna *ciutat* dins la família *dades_personals*, i en canvi, en la família *dades_professionals* afegeix dues columnes més: *antiguitat* i *numero_seguretat_social*. Això dona molta flexibilitat!

Amb l'ordre *scan*, podem llegir una taula (en veurem els detalls més endavant):

```
hbase(main):020:0> scan 'empleat'
ROW
1
1
1
1
1 row(s) in 0.0730 seconds

COLUMN+CELL
column=dades_personals:ciutat, timestamp=1679509257649, value=Eivissa
column=dades_personals:nom, timestamp=1679509254088, value=Joan
column=dades_professionals:departament, timestamp=1679509261883, value=Sistemes
column=dades_professionals:salari, timestamp=1679509265009, value=35000
```

Imatge: Lectura de la taula després d'haver inserit una fila

Podem veure que ha inserit una fila, amb 4 cel·les. Per a cada cel·la guarda un *timestamp* del moment en què s'ha inserit.

Vegem un *scan* després d'inserir les 3 files de la taula:

```

hbase(main):030:0> scan 'empleat'
ROW
1
1
1
1
2
2
2
2
3
3
3
3
3
3 row(s) in 0.0260 seconds

COLUMN+CELL
column=dades_personals:ciutat, timestamp=1679509257649, value=Eivissa
column=dades_personals:nom, timestamp=1679509254088, value=Joan
column=dades_professionals:departament, timestamp=1679509261883, value=Sistemes
column=dades_professionals:salari, timestamp=1679509265009, value=35000
column=dades_personals:ciutat, timestamp=1679509560451, value=Palma
column=dades_personals:nom, timestamp=1679509549468, value=Aina
column=dades_professionals:departament, timestamp=1679509563504, value=Desenvolupament
column=dades_professionals:salari, timestamp=1679509567565, value=40000
column=dades_personals:ciutat, timestamp=1679510010381, value=Ma\xC3\xB3
column=dades_personals:nom, timestamp=1679510007464, value=Pep
column=dades_professionals:departament, timestamp=1679510013212, value=Desenvolupament
column=dades_professionals:salari, timestamp=1679510016831, value=30000

```

Imatge: Lectura de la taula després d'haver inserit les tres files

Com veim, tenim 3 files i 12 cel·les inserides.

L'ordre *put* també serveix per modificar un valor d'una cel·la. Simplement utilitzam la mateixa sintaxi que per a la inserció i el valor que hi havia es sobreescriurà pel nou. Suposem que volem canviar la ciutat d'Aina perquè sigui Inca en lloc de Palma. Ho fariem així:

```
put 'empleat', '2', 'dades_personals:ciutat', 'Inca'
```

Llegir les dades d'una fila: get

L'ordre **get** permet llegir les dades d'una fila a partir de la seva clau. La seva sintaxi és:

```
get 'taula', 'clau de fila'
```

Per exemple, anam a veure les dades de l'empleat 1:

```
get 'empleat', '1'
```

Vegem que això retorna totes les cel·les associades a la fila 1:

```

hbase(main):031:0> get 'empleat', '1'
COLUMN                                CELL
dades_personals:ciutat                timestamp=1679509257649, value=Eivissa
dades_personals:nom                   timestamp=1679509254088, value=Joan
dades_professionals:departament       timestamp=1679509261883, value=Sistemes
dades_professionals:salari            timestamp=1679509265009, value=35000
4 row(s) in 0.0140 seconds

```

Imatge: Lectura de la fila 1

Esborrar dades: delete i deleteall

L'ordre **delete** esborra una cel·la concreta. Hem d'especificar la taula, fila, columna i, opcionalment, el seu timestamp:

```
delete 'taula', 'clau de fila', 'familia:columna' [, 'timestamp']
```

Per exemple, anam a esborrar el salari de Pep:

```
delete 'empleat', '3', 'dades_professionals:salari'
```

Si el que volem és esborrar una fila sencera, és a dir, totes les seves cel·les, hem d'emprar l'ordre **deleteall**. Anem a esborrar a l'empleada Aina:

```
deleteall 'empleat', '2'
```

Comptar files i truncar una taula: count i truncate

L'ordre **count** ens retorna el número de files d'una taula, 2 en el nostre cas, després d'haver esborrat un empleat:

```
hbase(main):023:0> count 'empleat'
2 row(s) in 0.0270 seconds

=> 2
```

Imatge: *Número de files d'una taula*

Per altra banda, l'ordre **truncate** trunca una taula, és a dir esborra totes les seves files. En realitat, internament el que fa és que desactiva la taula, l'esborra i la torna a crear. Ho podem veure en la imatge següent:

```
hbase(main):046:0> truncate 'empleat'
Truncating 'empleat' table (it may take a while):
- Disabling table...
- Truncating table...
0 row(s) in 3.7860 seconds
```

Imatge: *Truncat d'una taula*

Llegir dades d'una taula: scan

Ja hem vist anteriorment que amb l'ordre **scan** podem llegir tota una taula. Després de tornar a inserir les dades dels tres empleats a la nostra taula, amb la següent ordre podem veure els continguts de la taula:

```
scan 'empleat'
```

Recordem que el que mostra són totes les cel·les de cada una de les files de la taula.

```
hbase(main):030:0> scan 'empleat'
ROW
1 column=dades_personals:ciutat, timestamp=1679509257649, value=Eivissa
1 column=dades_personals:nom, timestamp=1679509254088, value=Joan
1 column=dades_professionals:departament, timestamp=1679509261883, value=Sistemes
1 column=dades_professionals:salari, timestamp=1679509265009, value=35000
2 column=dades_personals:ciutat, timestamp=1679509560451, value=Palma
2 column=dades_personals:nom, timestamp=1679509549468, value=Aina
2 column=dades_professionals:departament, timestamp=1679509563504, value=Desenvolupament
3 column=dades_professionals:salari, timestamp=1679509567565, value=40000
3 column=dades_personals:ciutat, timestamp=1679510010381, value=MaXc3Xx3
3 column=dades_personals:nom, timestamp=1679510007464, value=Pep
3 column=dades_professionals:departament, timestamp=1679510013212, value=Desenvolupament
3 column=dades_professionals:salari, timestamp=1679510016831, value=30000
3 row(s) in 0.0260 seconds
```

Imatge: *Scan de la taula*

L'ordre scan permet especificar un límit de files que es retornen (2 en el següent exemple):

```
scan 'empleat', {'LIMIT' => 2}
```

```
hbase(main):041:0> scan 'empleat', {'LIMIT' => 2}
ROW                                COLUMN+CELL
1                                  column=dades_personals:ciutat, timestamp=1679558727295, value=Eivissa
1                                  column=dades_personals:nom, timestamp=1679558727267, value=Joan
1                                  column=dades_professionals:departament, timestamp=1679558727326, value=Sistemes
1                                  column=dades_professionals:salari, timestamp=1679558728761, value=35000
2                                  column=dades_personals:ciutat, timestamp=1679558733747, value=Palma
2                                  column=dades_personals:nom, timestamp=1679558733708, value=Aina
2                                  column=dades_professionals:departament, timestamp=1679558733773, value=Desenvolupament
2                                  column=dades_professionals:salari, timestamp=1679558733803, value=40000
2 row(s) in 0.0330 seconds
```

Imatge: *Scan amb limitació de files*

I també podem especificar que només es retornin les dades de les columnes especificades, per exemple el nom i el departament:

[illegible]

```
hbase(main):021:0> scan 'empleat', {'COLUMNS' => ['dades_personals:nom','dades_professionals:departament']}
ROW                                COLUMN+CELL
1                                column=dades_personals:nom, timestamp=1679556152240, value=Joan
1                                column=dades_professionals:departament, timestamp=1679556159957, value=Sistemes
2                                column=dades_personals:nom, timestamp=1679556166267, value=Aina
2                                column=dades_professionals:departament, timestamp=1679556173534, value=Desenvolupament
3                                column=dades_personals:nom, timestamp=1679556179053, value=Pep
3                                column=dades_professionals:departament, timestamp=1679556185539, value=Desenvolupament
3 row(s) in 0.0190 seconds
```

Imatge: Scan amb limitació de columnes

També podem especificar que es recuperin només les cel·les amb un rang temporal (recordau que cada cel·la té un *timestamp*). Per exemple:

```
scan 't1', {COLUMNS => 'c1', TIMERANGE => [1303668804, 1303668904]}
```

```
hbase(main):072:0> scan 'empleat', {TIMERANGE => [1679558727295, 1679558733747]}
ROW
1      COLUMN+CELL
1      column=dades_personals:ciutat, timestamp=1679558727295, value=Eivissa
1      column=dades_professionals:departament, timestamp=1679558727326, value=Systemes
1      column=dades_professionals:salari, timestamp=1679558728761, value=35000
2      column=dades_personals:nom, timestamp=1679558733708, value=Aina
2 row(s) in 0.0130 seconds
```

Imatge: Scan amb limitació temporal

Per últim, el que també podem especificar són filtres, condicions que han de satisfer les dades que es retornin, l'equivalent a la clàusula WHERE d'un SELECT de SQL. Això ho veurem en detall en el següent subapartat.

7.1. Filtres

L'ordre scan permet definir filtres, de manera que només es recuperin les files que compleixin una condició. És l'equivalent a un WHERE de SQL. Però la manera de definir un filtre és una mica més complexa que en SQL. Normalment els filtres s'utilitzen amb l'API Java, però també poden emprar-se des del HBase Shell, que és el que veurem aquí.

Vegem primer quins són els filtres que tenim disponibles mitjançant l'ordre **show_filters**:

```
hbase(main):042:0> show_filters
ColumnPrefixFilter
TimestampsFilter
PageFilter
MultipleColumnPrefixFilter
FamilyFilter
ColumnPaginationFilter
SingleColumnValueFilter
RowFilter
QualifierFilter
ColumnRangeFilter
ValueFilter
PrefixFilter
SingleColumnValueExcludeFilter
ColumnCountGetFilter
InclusiveStopFilter
DependentColumnFilter
FirstKeyOnlyFilter
KeyOnlyFilter
```

Imatge: Filtres disponibles

La sintaxi d'un scan amb filtre és la següent:

```
scan 'empleat', {FILTER => "definició_del_filtre"}
```

A continuació veurem com es defineixen els filtres més utilitzats.

RowFilter

El filtre **RowFilter** recupera cel·les a partir de la clau de fila. Per exemple, si volem recuperar les cel·les que tenen clau de fila 1, ho farem així:

```
RowFilter(=, 'binary:1')
```

El primer paràmetre és un operador de comparació, que en aquest cas hem utilitzat el d'igualtat. Podríem emprar qualsevol altre: !=, <, <=, >, >=. El segon paràmetre és la clau de fila en bytes, per això empram el prefix **binary:**, perquè ho converteixi a bytes.

Vegem com executam un scan amb aquest filtre:

```
scan 'empleat', {FILTER => "RowFilter(=, 'binary:1')"}>
```

```
hbase(main):074:0> scan 'empleat', {FILTER => "RowFilter(=, 'binary:1')"}
ROW COLUMN+CELL
1 column=dades_personals:ciutat, timestamp=1679558727295, value=Eivissa
1 column=dades_personals:nom, timestamp=1679558727267, value=Joan
1 column=dades_professionals:departament, timestamp=1679558727326, value=Sistemes
1 column=dades_professionals:salari, timestamp=1679558728761, value=35000
1 row(s) in 0.0260 seconds
```

Imatge: Scan amb filtre RowFilter

ValueFilter

El filtre **ValueFilter** ens retorna les cel·les tals que el seu valor satisfà una condició. Per exemple, volem recuperar les cel·les tals que el seu valor és Aina. El filtre l'escriuríem així:

```
ValueFilter(=, 'binary:Aina')
```

Igual que en *RowFilter*, el primer paràmetre és un operador de comparació, mentre que el segon és el valor que cerquem en bytes. Vegem l'operació de *scan* amb aquest filtre:

```
scan 'empleat', {FILTER => "ValueFilter(=, 'binary:Aina')"}

```

```
hbase(main):006:0> scan 'empleat', {FILTER => "ValueFilter(=, 'binary:Aina')"}
ROW                                COLUMN+CELL
 2                                column=dades_personals:nom, timestamp=1679558733708, value=Aina
1 row(s) in 0.0110 seconds
```

Imatge: Scan amb filtre ValueFilter

Podem veure que ha retornat la cel·la (no la fila sencera) que té el valor Aina.

Podem també cercar una subcadena, emprant el prefix **substring**: en lloc de *binary*:. Per exemple, anam a cercar la subcadena 'me' en totes les cel·les:

```
scan 'empleat', {FILTER => "ValueFilter(=, 'substring:me')"}

```

El resultat són les tres cel·les que contenen 'me', en aquest cas dins els valors 'Sistemes' i 'Desenvolupament':

```
hbase(main):007:0> scan 'empleat', {FILTER => "ValueFilter(=, 'substring:me')"}
ROW                                COLUMN+CELL
 1                                column=dades_professionals:departament, timestamp=1679558727326, value=Sistemes
 2                                column=dades_professionals:departament, timestamp=1679558733773, value=Desenvolupament
 3                                column=dades_professionals:departament, timestamp=1679558739724, value=Desenvolupament
3 row(s) in 0.0240 seconds
```

Imatge: Scan amb filtre ValueFilter i substring:

I també és possible cercar fent servir una expressió regular, emprant el prefix **regexstring**:. Per exemple, volem recuperar les cel·les tals que el seu valor comencin per 'P':

```
scan 'empleat', {FILTER => "ValueFilter(=, 'regexstring:^P.*')"}

```

Podem comprovar que ens ha retornat dues cel·les, la de Palma i la de Pep:

```
hbase(main):012:0> scan 'empleat', {FILTER => "ValueFilter(=, 'regexstring:^P.*')"}
ROW                                COLUMN+CELL
 2                                column=dades_personals:ciutat, timestamp=1679558733747, value=Palma
 3                                column=dades_personals:nom, timestamp=1679558739666, value=Pep
2 row(s) in 0.0150 seconds
```

Imatge: Scan amb filtre ValueFilter i regexstring:

Per últim també tenim un altre prefix, **binaryprefix**:, que serveix per recuperar els valors que comencen per una determinada cadena. Per exemple, els que comencin per 'Pa':

```
scan 'empleat', {FILTER => "ValueFilter(=, 'binaryprefix:Pa')"}

```

```
hbase(main):016:0> scan 'empleat', {FILTER => "ValueFilter(=, 'binaryprefix:Pa')"}
ROW                                COLUMN+CELL
 2                                column=dades_personals:ciutat, timestamp=1679558733747, value=Palma
1 row(s) in 0.0320 seconds
```

Imatge: Scan amb filtre ValueFilter i binaryprefix:

Aquí hem vist els prefixs *binary*:, *substring*:, *regexstring*: i *binaryprefix*: associats al filtre *ValueFilter*. Però també poden emprar-se en qualsevol filtre que requereixi una comparació de valors. Per exemple, en el filtre *RowFilter* que hem vist abans (hi vàrem emprar *binary*: en l'exemple) o en el *SingleColumnValueFilter* que veurem a continuació.

I sempre poden emprar-se en combinació amb qualsevol dels operadors de comparació: =, !=, <, <=, >, >=

SingleColumnValueFilter

A diferència de *ValueFilter*, el filtre ***SingleColumnValueFilter*** ens permet especificar a quina columna volem cercar un valor. A més, aquest filtre no retorna cel·les aïllades, sinó totes les d'una fila que satisfaci la consulta. La sintaxi és la següent:

```
SingleColumnValueFilter('família_de_columnes','columna',operador_de_comparació,valor)
```

Per exemple, si volem cercar el valor Joan a la columna *dades_personals:nom*, ho escriuríem així:

```
SingleColumnValueFilter('dades_personals','nom',='binary:Joan')
```

Vegem l'execució de l'operació de *scan* amb aquest filtre:

```
scan 'empleat', {FILTER => "SingleColumnValueFilter(
    'dades_personals','nom',='binary:Joan')"}

```

```
hbase(main):025:0> scan 'empleat', {FILTER => "SingleColumnValueFilter('dades_personals','nom',='binary:Joan')"}
ROW
1      COLUMN+CELL
1      column=dades_personals:ciutat, timestamp=1679558727295, value=Eivissa
1      column=dades_personals:nom, timestamp=1679558727267, value=Joan
1      column=dades_professionals:departament, timestamp=1679558727326, value=Sistemes
1      column=dades_professionals:salari, timestamp=1679558728761, value=35000
1 row(s) in 0.0270 seconds
```

Imatge: Scan amb filtre *SingleColumnValueFilter*

Com podem comprovar, no només recupera la cel·la del nom, sinó que recupera totes les cel·les de la fila.

Tractament dels valors absents

Imaginem que no s'ha especificat la columna *dades_professionals:salari* per a la fila 3 (Pep). Si executam la query següent que pretén recuperar els empleats amb un salari major que 36.000 euros

```
scan 'empleat', {FILTER => "SingleColumnValueFilter(
    'dades_professionals','salari',>,'binary:36000')"}

```

ens retornarà les cel·les de la fila 2 (Aina, que té un salari de 40.000 euros), però també les de la fila 3 (Pep), tot i que no té cap salari definit. És a dir, quan cerquem per una columna, ens recupera les files dels valors que satisfan la condició (Aina en l'exemple), i també les files que no tenen definida la columna (Pep en l'exemple): un valors absent.

No és el més lògic, però és el comportament per defecte del filtre *SingleColumnValueFilter* en HBase. Per canviar aquest comportament, hem d'utilitzar un altre constructor del filtre *SingleColumnValueFilter*, que afegeix dos paràmetres al final:

- *boolean filterIfMissing*
- *boolean latestVersionOnly*

El que ens interessa és el primer (*filterIfMissing*). Si el posam a *true*, no retornarà els valors absents. Per defecte, el seu valor és *false*.

Pel que fa al segon paràmetre (*latestVersionOnly*), serveix per quan treballem amb múltiples versions (per defecte només es permet una, però podem canviar la propietat *VERSIONS*, tal i com hem vist a l'apartat de DDL), que només en recuperi la darrera. El valor per defecte és *true*. En els exemples que feim (i en la tasca) no feim ús de les versions, així que ho podem deixar a *true* o *false* indistintament.

Per tant, si no volem recuperar els valors absents (que és el més habitual), l'ordre *scan* anterior hauria de ser:

```
scan 'empleat', {FILTER => "SingleColumnValueFilter(
    'dades_professionals', 'salari', '>', 'binary:36000', true, true)"}

```

I ara ja sí que recupera només les cel·les de la fila 2 (Aina, amb salari de 40.000 euros).

FamilyFilter

El filtre FamilyFilter és una mica diferent als anteriors ja que serveix per recuperar tots els valors per a les columnes d'una determinada família, però sense aplicar cap condició. Per exemple, si volem recuperar els valors de les dades personals, el filtre seria:

```
FamilyFilter(=, 'binary:dades_personals')
```

Si veim l'execució de l'operació de *scan*, podem comprovar que retorna els valors de les dues cel·les corresponent a la família de columnes per a cada una de les 3 files:

```
scan 'empleat', {FILTER => "FamilyFilter(=, 'binary:dades_personals')"}

```

```
hbase(main):028:0> scan 'empleat', {FILTER => "FamilyFilter (=, 'binary:dades_personals')"}
ROW                                COLUMN+CELL
1                                  column=dades_personals:ciutat, timestamp=1679558727295, value=Eivissa
1                                  column=dades_personals:nom, timestamp=1679558727267, value=Joan
2                                  column=dades_personals:ciutat, timestamp=1679558733747, value=Palma
2                                  column=dades_personals:nom, timestamp=1679558733708, value=Aina
3                                  column=dades_personals:ciutat, timestamp=1679558739704, value=Ma\xC3\xB3
3                                  column=dades_personals:nom, timestamp=1679558739666, value=Pep
3 row(s) in 0.0240 seconds

```

Imatge: Scan amb filtre FamilyFilter

Operadors lògics

Podem utilitzar els operadors lògics **AND** i **OR** per combinar diferents filtres. Per exemple, si volem les files que tinguin un salari entre 33.000 i 36.000 euros, podem utilitzar dos filtres SingleColumnValueFilter i emprar l'operador AND:

```
scan 'empleat', {FILTER => "SingleColumnValueFilter(
    'dades_professionals', 'salari', '>=', 'binary:33000')
AND SingleColumnValueFilter('dades_professionals', 'salari', '<=', 'binary:36000')"}

```

```
hbase(main):034:0> scan 'empleat', {FILTER => "SingleColumnValueFilter('dades_professionals', 'salari', '>=', 'binary:33000') AND SingleColumnValueFilter('dades_professionals', 'salari', '<=', 'binary:36000')"}
ROW                                COLUMN+CELL
1                                  column=dades_personals:ciutat, timestamp=1679558727295, value=Eivissa
1                                  column=dades_personals:nom, timestamp=1679558727267, value=Joan
1                                  column=dades_professionals:departament, timestamp=1679558727326, value=Sistemes
1                                  column=dades_professionals:salari, timestamp=1679558728761, value=35000
1 row(s) in 0.0140 seconds

```

Imatge: Scan amb operador AND

Un altre exemple: volem els empleats que siguin de Palma o del departament de Sistemes. En aquest cas emprem OR:

```
scan 'empleat', {FILTER => "SingleColumnValueFilter(
    'dades_personals', 'ciutat', '=', 'binary:Palma')
OR SingleColumnValueFilter(
    'dades_professionals', 'departament', '=', 'binary:Sistemes')"}

```

```
hbase(main):042:0> scan 'empleat', {FILTER => "SingleColumnValueFilter('dades_personals', 'ciutat', '=', 'binary:Palma') OR SingleColumnValueFilter('dades_professionals', 'departament', '=', 'binary:Sistemes')"}
ROW                                COLUMN+CELL
1                                  column=dades_personals:ciutat, timestamp=1679558727295, value=Eivissa
1                                  column=dades_personals:nom, timestamp=1679558727267, value=Joan
1                                  column=dades_professionals:departament, timestamp=1679558727326, value=Sistemes
1                                  column=dades_professionals:salari, timestamp=1679558728761, value=35000
2                                  column=dades_personals:ciutat, timestamp=1679558733747, value=Palma
2                                  column=dades_personals:nom, timestamp=1679558733708, value=Aina
2                                  column=dades_professionals:departament, timestamp=1679558733773, value=Desenvolupament
2                                  column=dades_professionals:salari, timestamp=1679558733803, value=40000
2 row(s) in 0.0220 seconds

```

Imatge: Scan amb operador OR

Per acabar, podem emprar aquests operadors per combinar diferents filtres. Per exemple, volem recuperar les files del departament de Desenvolupament amb un SingleColumnValueFilter i aplicar després un FamilyFilter per recuperar només les cel·les de la família de columnes de dades personals:

```
scan 'empleat', {FILTER => "SingleColumnValueFilter(
'dades_professionals','departament',=,'binary:Desenvolupament')
AND FamilyFilter(=,'binary:dades_personals')"}}
```

```
hbase(main):044:0> scan 'empleat', {FILTER => "SingleColumnValueFilter('dades_professionals','departament',=,'binary:Desenvolupament') AND FamilyFilter(=,'binary:dades_personals')"}
ROW                                COLUMN+CELL
2                                  column=dades_personals:ciutat, timestamp=1679558733747, value=Palma
2                                  column=dades_personals:nom, timestamp=1679558733788, value=Aina
3                                  column=dades_personals:ciutat, timestamp=1679558739704, value=Ma\xC3\xB3
3                                  column=dades_personals:nom, timestamp=1679558739666, value=Pep
2 row(s) in 0.0300 seconds
```

Imatge: Scan amb diferents filtres



AMPLIACIÓ

A la documentació de la darrera versió de Cloudera podem trobar els detalls de tots els filtres suportats per la versió que incorpora d'HBase i que són els mateixos que tenim en la nostra màquina virtual de Cloudera QuickStart.

Ho teniu a: <https://docs.cloudera.com/runtime/7.2.17/managing-hbase/topics/hbase-filtering.html>

7.2. Inserció de fitxers CSV

En l'apartat 5 hem vist que la interfície web de Hue ens permet pujar un arxiu amb dades. Molt sovint, empram aquest mètode per inserir dades, perquè fer-ho mitjançant ordres put es fa molt feixuc.

L'arxiu ha de ser en format CSV i a la primera línia ha de contenir els noms de les columnes, amb el seu nom de família. A més, la clau de fila ha d'anar en una primera columna, amb el nom `:key`.

Per exemple, aquest seria un fitxer CSV que podríem pujar i que conté dos empleats més per a la nostra taula:

```
:key,dades_personals:nom,dades_personals:ciutat,dades_professionals:departament,dades_professionals:salari
4,Jaume,Manacor,Sistemes,32000
5,Marga,Inca,Desenvolupament,42000
```

Una altra manera d'importar fitxers CSV a una taula ja creada és amb l'eina **ImportTsv**. Primer hem de tenir el nostre fitxer CSV dins HDFS. Suposem que el tenim en el path `/hbase/empleats.csv`. És important que si empram aquesta eina el fitxer no tenguí cap capçalera, perquè en cas contrari, la ficarà com una fila més. Així doncs, el nostre fitxer CSV ara serà:

```
4,Jaume,Manacor,Sistemes,32000
5,Marga,Inca,Desenvolupament,42000
```

Ara, des de la consola, hem d'escriure la següent ordre (tot en una línia i sense deixar cap espai en la llista de columnes):

```
hbase org.apache.hadoop.hbase.mapreduce.ImportTsv
-Dimporttsv.separator=,
-Dimporttsv.columns="HBASE_ROW_KEY,
dades_personals:nom, dades_personals:ciutat,
dades_professionals:departament, dades_professionals:salari"
empleat
/hbase/empleats.csv
```

On amb `-Dimporttsv.separator` indicam que utilitzam comes per separar els camps del fitxer i amb `-Dimporttsv.columns` posam, entre cometes, la llista de columnes de la taula HBase en l'ordre en què apareixen en el fitxer i on la clau es representa amb `HBASE_ROW_KEY`. A continuació indicam el nom de la taula HBase (`empleat`) i finalment el path HDFS del fitxer CSV que importam (`/hbase/empleats.csv`).

Veureu que quan s'executi l'ordre es llançarà un treball mapreduce que acaba amb la inserció de les dades en la taula `empleat`.

8. HBase i Hive

Recordem del lliurament anterior del mòdul de Big data aplicat que Apache Hive es una infraestructura de codi obert per a *data warehousing*, que crea una capa que permet accedir a les dades massives emmagatzemades en un clúster Hadoop mitjançant un llenguatge molt semblant a SQL, anomenat HiveQL. Les dades estan normalment emmagatzemades en HDFS, però també poden estar-ho en HBase.

A continuació anam a crear una nova taula en Hive que volem que s'emmagatzemi dins HBase. La taula en Hive es dirà *hbase_empleat2* i tindrà la mateixa definició que la taula *empleat* amb la qual hem treballat en els apartats anteriors.

```
CREATE TABLE hbase_empleat2(key bigint, nom string, ciutat string,
    departament string, salari int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,
    dades_personals:nom, dades_personals:ciutat,
    dades_professionals:departament, dades_professionals:salari")
TBLPROPERTIES ("hbase.table.name" = "empleat2");
```

Veim que hem definit la taula *hbase_empleat2* de Hive amb les columnes que necessitam, incloent una per a la clau. Amb *STORE BY* especifiquem que la taula s'emmagatzemarà dins HBase. Dins les *SERDEPROPERTIES* especifiquem els *mappings*, per ordre, entre les columnes de la taula de Hive i les columnes de la taula d'HBase, amb les seves famílies. És important incloure la clau aquí. Per últim, acabam dient quin serà el nom de la taula en HBase: *empleat2*.

Això crea la taula *empleat2* en HBase, amb una clau i dues famílies de columnes, amb dues columnes cadascuna. Si ja existís una taula *empleat2* dins HBase, donaria un error. A més, crea la taula *hbase_empleat2* en Hive, és a dir, afegeix les metadades necessàries en el metastore de Hive per poder usar-la.

A partir d'aquí ja podem treballar amb la taula *hbase_empleat2* dins Hive com amb qualsevol taula. Podríem carregar un fitxer amb un *LOAD* o fer *inserts* directament. Per exemple:

```
insert into hbase_empleat2 values(1, 'Joan', 'Eivissa', 'Sistemes', 35000);
```

Quan l'executem, veurem que es genera i llença un treball mapreduce que insereix una fila en la taula HBase.

Ara podem comprovar que la fila està dins HBase amb un *scan*:

```
hbase(main):007:0> scan 'empleat2'
ROW
1
1
1
1
1
1 row(s) in 0.0150 seconds

COLUMN+CELL
column=dades_personals:ciutat, timestamp=1679583381469, value=Eivissa
column=dades_personals:nom, timestamp=1679583381469, value=Joan
column=dades_professionals:departament, timestamp=1679583381469, value=Sistemes
column=dades_professionals:salari, timestamp=1679583381469, value=35000
```

Imatge: Scan de la taula *empleat2* en HBase

I també podem comprovar-ho des de Hive amb un *select*:

```
0: jdbc:hive2://> select * from hbase_empleat2;
OK
+-----+-----+-----+-----+-----+
| hbase_empleat2.key | hbase_empleat2.nom | hbase_empleat2.ciutat | hbase_empleat2.departament | hbase_empleat2.salari |
+-----+-----+-----+-----+-----+
| 1 | Joan | Eivissa | Sistemes | 35000 |
+-----+-----+-----+-----+-----+
1 row selected (0.259 seconds)
```

Imatge: Select de la taula *hbase_empleat2* en Hive

Aquesta és una taula "normal" (interna) de Hive, la única diferència amb les que vàrem treballar en el lliurament anterior és que aquí s'emmagatzema dins HBase. Si des de Hive esborram la taula amb un *drop table*, automàticament també s'esborrarà de HBase.

Taules externes de Hive

Encara és més interessant i habitual treballar amb taules externes de Hive. Recordem que una taula externa vol dir que es gestiona externament a Hive, de manera que el que tenim és només una espècie d'enllaç (unes metadades al catàleg). D'aquesta manera, quan esborram una taula externa, no esborram les dades, només les metadades al catàleg.

Això ens permet que si ja tenim una taula creada en HBase (en el nostre cas, la taula *empleat* amb la qual hem treballat en els apartats anteriors), la puguem definir com una taula externa dins Hive. La sintaxi és pràcticament idèntica a la que hem vist abans, però afegint la paraula **EXTERNAL**:

```
CREATE EXTERNAL TABLE hbase_empleat(key bigint, nom string, ciutat string,
departament string, salari int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,
dades_personals:nom, dades_personals:ciutat,
dades_professionals:departament, dades_professionals:salari")
TBLPROPERTIES("hbase.table.name" = "empleat");
```

Aquí la taula *empleat* ja ha d'existir dins HBase, en cas contrari donaria un error. I simplement afegirà les metadades al metastore de Hive per poder emprar-la des de Hive. Així que ja podem inserir una nova fila:

```
insert into hbase_empleat values(4,'Jaume','Manacor','Sistemes',32000);
```

Igual que abans, es genera i llença el treball mapreduce que insereix la nova fila a la taula *empleat* en HBase. Ho podem comprovar amb un **get** en HBase:

```
hbase(main):009:0> get 'empleat','4'
COLUMN                                CELL
dades_personals:ciutat                timestamp=1679584638332, value=Manacor
dades_personals:nom                   timestamp=1679584638332, value=Jaume
dades_professionals:departament       timestamp=1679584638332, value=Sistemes
dades_professionals:salari            timestamp=1679584638332, value=32000
4 row(s) in 0.0210 seconds
```

Imatge: Get de la nova fila (4) en HBase

I també amb un **select** en Hive:

```
0: jdbc:hive2://> select * from hbase_empleat;
OK
+-----+-----+-----+-----+-----+
| hbase_empleat.key | hbase_empleat.nom | hbase_empleat.ciutat | hbase_empleat.departament | hbase_empleat.salari |
+-----+-----+-----+-----+-----+
| 1 | Joan | Eivissa | Sistemes | 35000 |
| 2 | Aina | Palma | Desenvolupament | 40000 |
| 3 | Pep | Maó | Desenvolupament | 30000 |
| 4 | Jaume | Manacor | Sistemes | 32000 |
+-----+-----+-----+-----+-----+
4 rows selected (0.296 seconds)
```

Imatge: Select de la taula *hbase_empleat* en Hive

Podeu comprovar que si ara esborram la taula en Hive amb un *drop table*, només s'esborra dins Hive (les seves metadades), però la taula *empleat* no es modifica dins HBase.



HBase ens proporciona un emmagatzematge capaç de treballar amb grans taules, amb una definició de dades molt flexible, d'una manera eficient i distribuïda.

Hive ens ofereix una manera més senzilla de consultar les dades, mitjançant sentències *SELECT*.

Resumint, Hive i HBase son dues eines que treballen molt bé juntes!



Podeu trobar més informació sobre la integració de Hive i HBase a la documentació de Hive:

<https://cwiki.apache.org/confluence/display/hive/hbaseintegration>

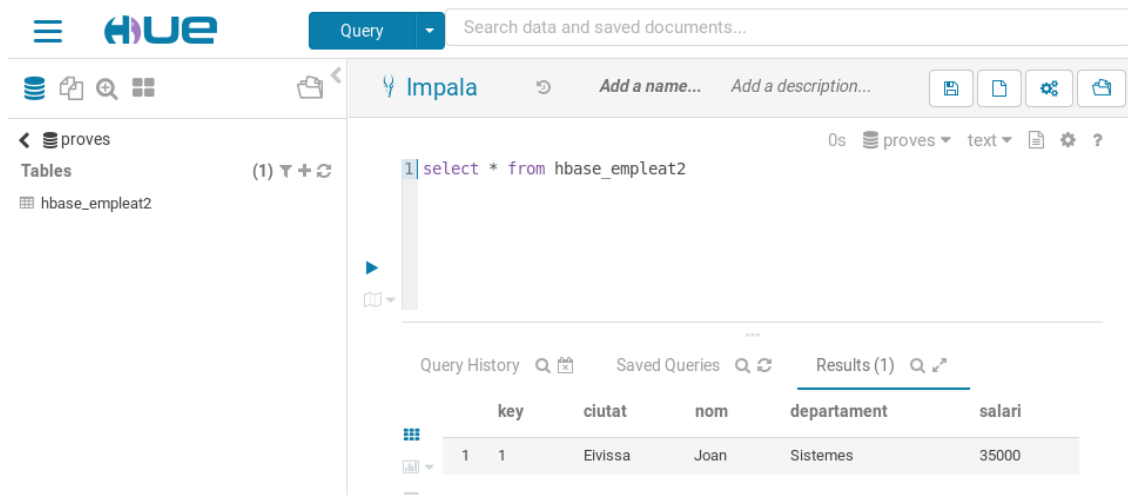
9. HBase i Impala

En l'apartat anterior hem vist com HBase i Hive treballen molt bé junts. I ja sabem que Impala (que estem veient en aquest lliurament del mòdul de Big data aplicat) està molt relacionat amb Hive, ja que fa servir el mateix catàleg de metadades (*metastore*) que Hive. Per tant, també podem treballar amb Impala per executar consultes SQL eficients sobre taules HBase.

Així doncs, la part de definició de la taula la farem des de Hive, ja sigui des del client Shell (*beeline*) o des de Hue. És probable que inicialment la nova taula no sigui visible des d'Impala i sigui necessari actualitzar el catàleg d'Impala. Per fer-ho, en l'editor d'Impala de Hue podem pitjar la icona de recarregar (*Refresh*) en el llistat de taules de la base de dades, i a continuació seleccionar l'opció "*Invalidate all metadata and rebuild index*". També podem fer-ho des del Shell d'Impala executant l'ordre:

```
INVALIDATE METADATA;
```

Finalment, ja podem executar la nostra consulta SQL des del client Impala, ja sigui Hue o *impala-shell*.



The screenshot shows the Hue web interface for Impala. The query editor contains the SQL statement: `select * from hbase_empleat2`. The results are displayed in a table with the following columns: **key**, **ciutat**, **nom**, **departament**, and **salari**. The results table shows one row of data:

key	ciutat	nom	departament	salari
1	Eivissa	Joan	Sistemes	35000

Imatge: Consulta amb Impala sobre una taula HBase

AMPLIACIÓ

Podeu trobar més informació sobre la utilització d'Impala per a executar consultes sobre taules HBase a la documentació d'Impala: https://impala.apache.org/docs/build/html/topics/impala_hbase.html