

# Hive

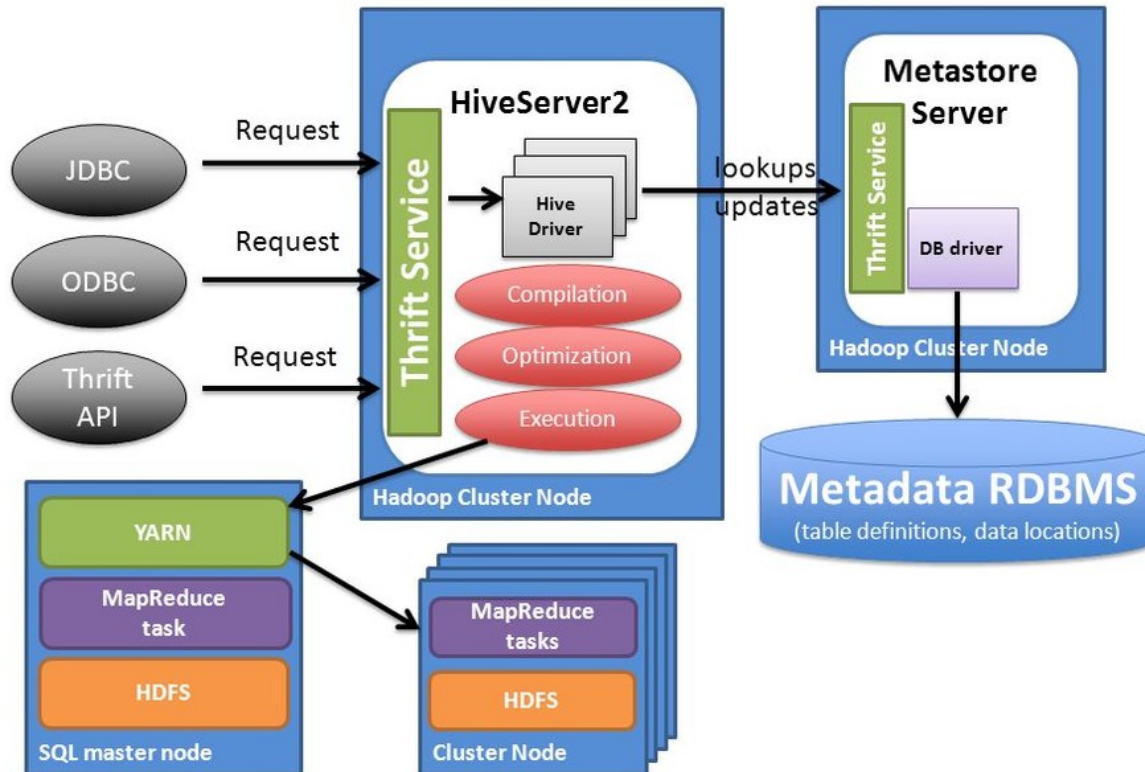


# Plan

- Architecture / intégration dans l'écosystème Hadoop
  - i. Qu'est-ce qu'un métastore ?
  - ii. Que se passe-t-il lors de l'exécution de requêtes SQL en Hive
- CRUD : Create Read Update Delete
  - i. Création d'une base
  - ii. Création d'une table (lecture locale, lecture HDFS)
  - iii. Requêtes
- Pourquoi Hive n'est pas un SGDB ?
- Spécificités :
  - i. Partitions
  - ii. Subqueries
  - iii. Clustering
  - iv. Tables externes VS internes

# Hive architecture générale

## Hive Architecture



# Présentation du client graphique

Hors précision du contraire, l'ensemble des manips qui vont suivre se feront sur le client graphique développé par HortonWorks.

Celui-ci se trouve en **port 30800**



**Data Analytics Studio**

# Présentation du client graphique

The screenshot shows the Data Analytics Studio interface in a Mozilla Firefox browser. The browser tabs include YouTube, Floor Plan Generation, Hortonworks Sandbox, Ambari - Sandbox, YARN, Data Analytics Studio, and hue - Recherche Google. The address bar shows the URL: sandbox-hdp.hortonworks.com:30800/#/databases/test/tables. The page title is "Database". The Database Explorer shows "LAST UPDATE: 705221 sec ago" and "test". The "No Tables Available." message is displayed. A modal window titled "Configurations" is open, showing the following details:

| Property                 | Value  |
|--------------------------|--|
| User                     | hive   |
| Cluster Id               | 23a01a6d-d837-4dc0-9f46-cb05ebb7c9da   |
| Database product         | Apache Hive  |
| Database product version | 3.1.0.3.0.1.0-187  |
| Product name             | DATA ANALYTICS STUDIO  |
| Product version          | 1.0.2.1.0.2.0-6  |
| JDBC connection          | jdbc:hive2://sandbox-hdp.hortonworks.com:2181/?serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2 |

A green "OK" button is visible at the bottom right of the modal window. Handwritten annotations include the word "about" in the top right and a large arrow pointing from the top right towards the modal window.

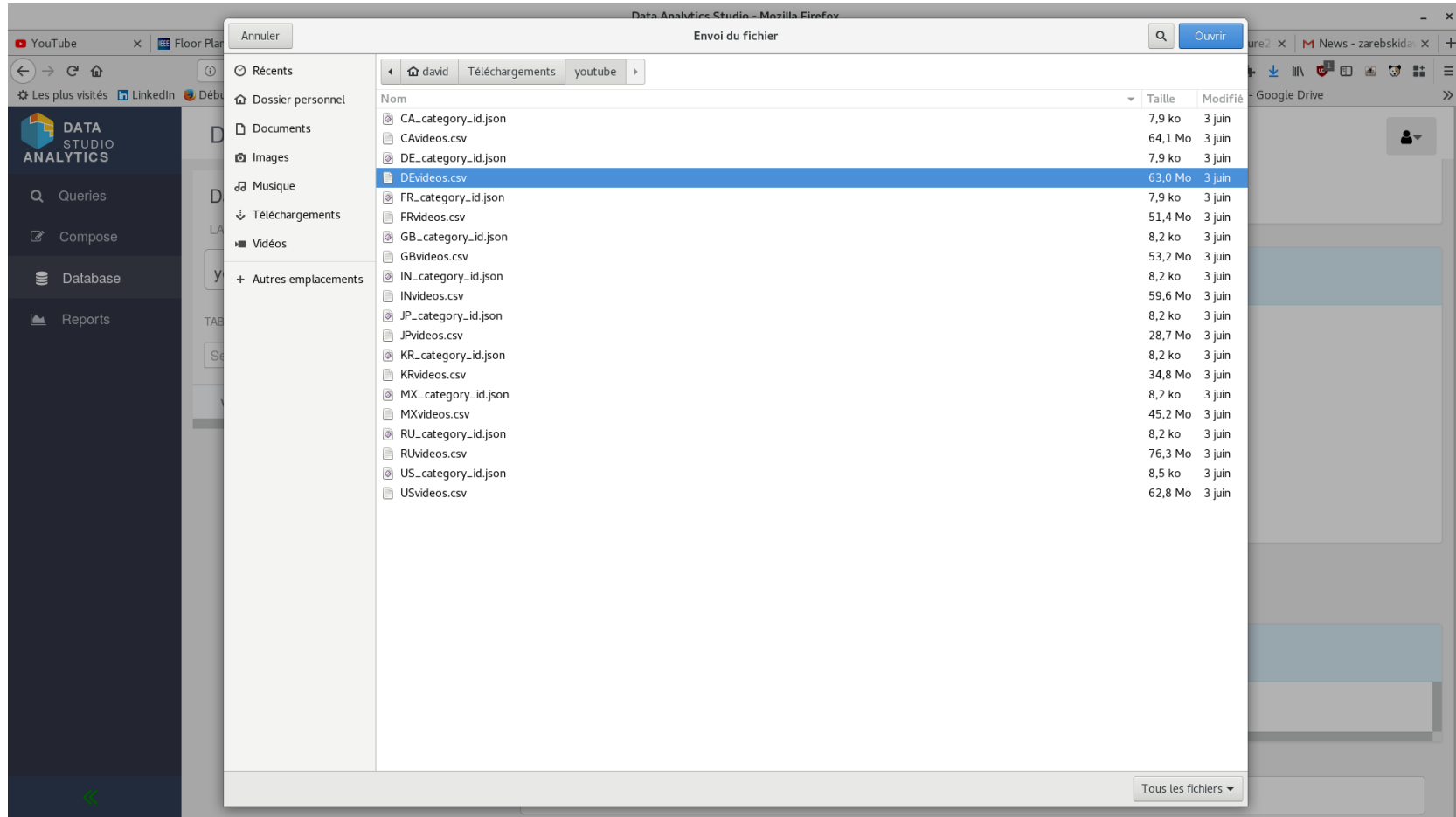
# CRUD : création d'une BDD

The screenshot displays the Data Analytics Studio web application in a Mozilla Firefox browser. The browser's address bar shows the URL: `sandbox-hdp.hortonworks.com:30800/#/databases/test/tables/new-database`. The interface features a dark sidebar on the left with navigation options: 'Queries', 'Compose', 'Database' (highlighted with a red box), and 'Reports'. The main content area is titled 'Database / NewDatabase' and includes a '+ Create Database' button at the top. Below this, a 'Database Explorer' panel on the left shows a tree structure with 'test' as the selected database, containing 'TABLES | 0'. A red arrow points from the 'Database Explorer' panel to the '+ Create Database' button. The main area contains a form for 'New Database Name' with a text input field labeled 'Database Name' and two buttons: 'CREATE' (orange) and 'CANCEL' (blue). At the bottom of the main area, a message states 'No Tables Available.'

# Création d'une table par import local

The screenshot shows the Data Analytics Studio interface in Mozilla Firefox. The browser tabs include YouTube, Floor Plan Generator, Hortonworks Sandbox, Ambari - Sandbox, Data Analytics Studio, hue - Recherche, data studio analytics, hive\_architecture2, News - zarebskida, and others. The address bar shows the URL: sandbox-hdp.hortonworks.com:30800/#/databases/youtube/tables/upload-table. The interface has a sidebar on the left with the 'DATA STUDIO ANALYTICS' logo and navigation options: Queries, Compose, Database, and Reports. The main content area is titled 'Database / UploadTable' and contains two panels: 'Select File Format' and 'Select File Source'. In the 'Select File Format' panel, the 'File type' is set to 'CSV', 'Field Delimiter' is a comma, 'Escape Character' is a backslash, and 'Quote Character' is a double quote. The 'Is first row header?' checkbox is checked, and 'Contains endlines?' is unchecked. In the 'Select File Source' panel, the 'Upload from Local' radio button is selected. A red circle highlights the 'Upload Table' button in the sidebar, and another red circle highlights the 'Upload from Local' radio button. A third red circle highlights the 'Select Local File' button at the bottom.

# Création d'une table par import local





# Création d'une table par import local

Data Analytics Studio - Mozilla Firefox

sandbox-hdp.hortonworks.com:30800/#/databases/youtube/tables/upload-table

Database / UploadTable

Database Explorer

LAST UPDATE: 7 sec ago

youtube

TABLES | 1

Search

video\_de

Name DEVideos

COLUMNS ADVANCED TABLE PROPERTIES

| COLUMN NAME   | DATA TYPE | SIZE | ADVANCED | ACTION   |
|---------------|-----------|------|----------|----------|
| video_id      | STRING    |      |          | ✕ DELETE |
| trending_date | STRING    |      |          | ✕ DELETE |
| title         | STRING    |      |          | ✕ DELETE |
| channel_title | STRING    |      |          | ✕ DELETE |
| category_id   | INT       |      |          | ✕ DELETE |
| publish_time  | STRING    |      |          | ✕ DELETE |
| tags          | STRING    |      |          | ✕ DELETE |
| views         | INT       |      |          | ✕ DELETE |

les types doivent être identifiés

# Renommer la table en videos\_de

Data Analytics Studio - Mozilla Firefox

sandbox-hdp.hortonworks.com:30800/#/databases/youtube/tables/video\_de/columns

Database / Table / Columns

Database Explorer

LAST UPDATE: 1 sec ago

youtube

TABLES | 1

Search

video\_de

TABLE > VIDEO\_DE

COLUMNS PARTITIONS STORAGE INFORMATION DETAILED INFORMATION

DATA PREVIEW

Search...

Search

ACTIONS : EDIT RENAME DELETE

| COLUMN NAME   | COLUMN TYPE | COMMENT        | CLUSTERED |
|---------------|-------------|----------------|-----------|
| video_id      | string      | Not Available! | false     |
| trending_date | string      | Not Available! | false     |
| title         | string      | Not Available! | false     |
| channel_title | string      | Not Available! | false     |
| category_id   | int         | Not Available! | false     |
| publish_time  | string      | Not Available! | false     |
| tags          | string      | Not Available! | false     |
| views         | int         | Not Available! | false     |
| likes         | int         | Not Available! | false     |
| dislikes      | int         | Not Available! | false     |

sandbox-hdp.hortonworks.com:30800/#/databases/youtube/tables/video\_de/rename

# Super, mais que c'est-il passé ?

Principalement, deux choses :

L'utilitaire graphique génère en réalité une **requête hql**. A l'exécution de cette dernière :

- Écriture des fichiers en HDFS
- Ecriture d'une entrée dans le **métastore**

# Allons voir cette requête

Data Analytics Studio - Mozilla Firefox

sandbox-hdp.hortonworks.com:30800/#/

hve architecture

Les plus visités LinkedIn Débuter avec Firefox From Google Chrome Entrepreneariat Informatique Google Agenda Wordreference Google Scholar pistes pro Linguee Dictionnaire a... Mon Drive - Google Drive

**DATA STUDIO ANALYTICS**

- Queries
- Compose
- Database
- Reports

## Queries

|   |         |      |                       |                 |  |  |
|---|---------|------|-----------------------|-----------------|--|--|
| ✓ select * from psuodirveehfbadphq...     | SUCCESS | hive | psuodirveehfbadphq... | Not Available!  |  |  |
| ✓ select * from devideos limit 20         | SUCCESS | hive | devideos (youtube)    | Not Available!  |  |  |
| ✓ SELECT * FROM psuodirveehfbadphq...     | SUCCESS | hive | psuodirveehfbadphq... | Not Available!  |  |  |
| ✓ SELECT * FROM devideos LIMIT 100        | SUCCESS | hive | devideos (youtube)    | Not Available!  |  |  |
| ✓ CREATE TABLE `youtube`.`psuodirveeh...  | SUCCESS | hive | Not Available!        | psuodirveehfba  |  |  |
| ✓ CREATE TABLE `youtube`.`DEvideos` ( ... | SUCCESS | hive | Not Available!        | DEvideos (youtu |  |  |
| ✓ DROP TABLE `youtube`.`pzqhwcbxyidjq...  | SUCCESS | hive | pzqhwcbxyidjqxyqew... | pzqhwcbxyidjqxy |  |  |
| ✓ DROP TABLE `youtube`.`devideos`         | SUCCESS | hive | devideos (youtube)    | devideos (youtu |  |  |
| ✓ SELECT * FROM pzqhwcbxyidjqxyqewa...    | SUCCESS | hive | pzqhwcbxyidjqxyqew... | Not Available!  |  |  |
| ✓ SELECT * FROM devideos LIMIT 100        | SUCCESS | hive | devideos (youtube)    | Not Available!  |  |  |
| ✓ CREATE TABLE `youtube`.`pzqhwcbxyid...  | SUCCESS | hive | Not Available!        | pzqhwcbxyidjqxy |  |  |
| ✓ CREATE TABLE `youtube`.`DEvideos` ( ... | SUCCESS | hive | Not Available!        | DEvideos (youtu |  |  |
| ✓ CREATE DATABASE `Youtube`               | SUCCESS | hive | Not Available!        | Not Available!  |  |  |
| ✓ select * from account limit 20          | SUCCESS | hive | account (foodmart)    | Not Available!  |  |  |

1 - 18 of 18 Show: 25 Go: 1

sandbox-hdp.hortonworks.com:30800/#/query/hive\_20190830121905\_22eedde8-4287-42d1-98c5-e1bac598e78e

# Allons voir cette requête

The screenshot shows the Data Analytics Studio interface in a Mozilla Firefox browser. The browser's address bar displays the URL: `sandbox-hdp.hortonworks.com:30800/#/query/hive_20190830121818_e9bef548-7632-477d-b613-35261c974d26`. The page title is "Queries / hive\_20190830121818\_e9bef548-7632-477d-b613-35261c974d26".

The interface features a left sidebar with the "DATA STUDIO ANALYTICS" logo and navigation links for "Queries", "Compose", "Database", and "Reports". The "Queries" link is selected.

The main content area is titled "Query Details" and contains the following information:

- QUERY ID:** hive\_20190830121818\_e9bef548-7632-477d-b613-35261c974d26
- USER:** hive
- STATUS:** ✓ SUCCESS
- START TIME:** 30 Aug 2019 14:18:18
- END TIME:** 30 Aug 2019 14:18:19
- DURATION:** 194ms
- TABLES READ:** Not Available!
- TABLES WRITTEN:** pzqhwcbxyidjxyqewatnradnzsohm (youtube)
- APPLICATION ID:** Not Available!

On the right side of the "Query Details" section, there are tabs for "RECOMMENDATIONS", "QUERY DETAILS" (which is active), "VISUAL EXPLAIN", "CONFIGS", "TIMELINE", and "DAG INFO". A green "DOWNLOAD" button is located to the right of these tabs.

The "QUERY DETAILS" tab displays the SQL query code in a text editor with line numbers 1 through 19:

```
1 CREATE TABLE `youtube`.`pzqhwcbxyidjxyqewatnradnzsohm` (  
2   `video_id` STRING,  
3   `trending_date` STRING,  
4   `title` STRING,  
5   `channel_title` STRING,  
6   `category_id` INT,  
7   `publish_time` STRING,  
8   `tags` STRING,  
9   `views` INT,  
10  `likes` INT,  
11  `dislikes` INT,  
12  `comment_count` INT,  
13  `thumbnail_link` STRING,  
14  `comments_disabled` BOOLEAN,  
15  `ratings_disabled` BOOLEAN,  
16  `video_error_or_removed` BOOLEAN,  
17  `description` STRING  
18 ) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' ESCAPED BY '|' STORED AS TEXTFILE  
19
```

# Où sont stockés les fichiers, exactement ?

A la suite de notre import de données locales, allez jeter un œil dans l'HDFS

```
/hive/managed/tablespace/warehouse/youtube.db/video_de
```

# Comment Hive sait-il où se trouvent les données et comment ces dernières sont structurés ?

Ces méta-données sont stockées dans le métastore (dans le cas présent, une DBB Mysql). Nous allons nous y connecter (le mdp est *hortonworks1*)

```
mysql -u root -p
```

```
use hive ;
```

# Comment Hive sait-il où se trouvent les données et comment ces dernières sont structurés ?

## Exercice

à l'aide d'une requête (et d'un peu de bon sens)

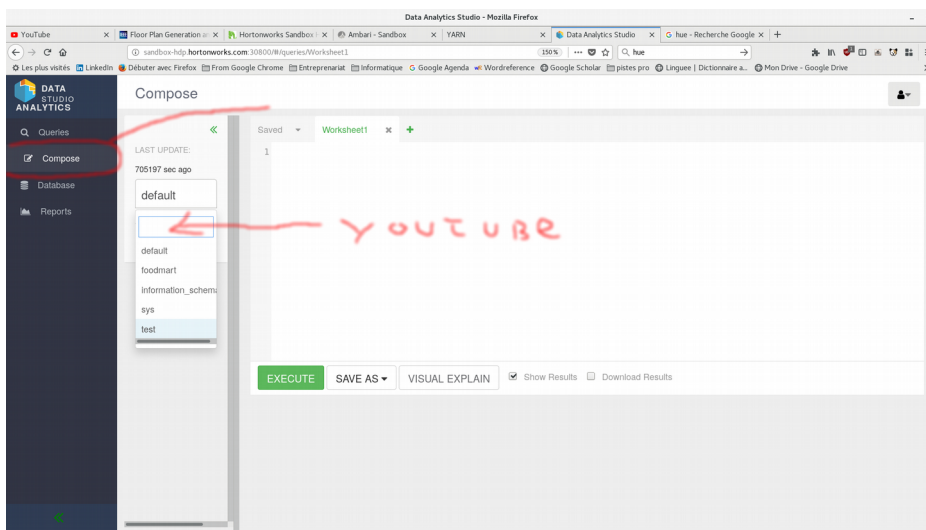
- Affichez l'ensemble des tables de la base de données youtube



# Requêtons sur la table

A l'aide d'une requête récupérez :

- l'id, somme des vues des vidéos ayant dépassé les 10000 visionnages au total (groupé par video\_id)



# Création d'une table à partir des données de l'HDFS

Nous allons cette fois aller plus loin que ce que nous permet l'assistant graphique.

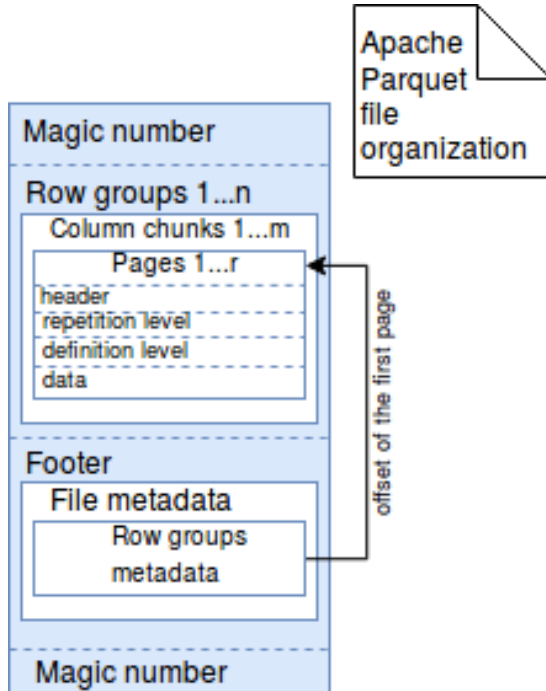
Nous allons créer une **table partitionnée** par lecture directe des fichiers sur l'HDFS.

Ceci suppose 2 queries :

- **CREATE TABLE**
- **LOAD DATA**

Mais commençons par convertir nos données dans un meilleur format : le format **Parquet**

# Mais, c'est quoi un Parquet ?












| Spark Format Showdown                     |                               | File Format        |                       |                     |
|---|-------------------------------|--------------------|-----------------------|---------------------|
|   |                               | CSV                | JSON                  | Parquet             |
| A<br>t<br>t<br>r<br>i<br>b<br>u<br>t<br>e | Columnar                      | No                 | No                    | Yes                 |
|   | Compressable                  | Yes                | Yes                   | Yes                 |
|   | Splittable                    | Yes*               | Yes**                 | Yes                 |
|   | Human Readable                | Yes                | Yes                   | No                  |
|   | Nestable                      | No                 | Yes                   | Yes                 |
|   | Complex Data Structures       | No                 | Yes                   | Yes                 |
|   | Default Schema: Named columns | Manual             | Automatic (full read) | Automatic (instant) |
|   | Default Schema: Data Types    | Manual (full read) | Automatic (full read) | Automatic (instant) |

# D'autres formats propres au Big Data

Même objectif de  
sérialisation avec un schéma  
explicite mais logiques  
différentes

BIG DATA FORMATS COMPARISON

|                           | Avro  | Parquet   | ORC   |
|---------------------------|---|---|---|
| Schema Evolution Support  |  |  |  |
| Compression               |  |  |  |
| Splitability              |  |  |  |
| Most Compatible Platforms | Kafka, Druid  | Impala, Arrow Drill, Spark  | Hive, Presto  |
| Row or Column             | Row   | Column  | Column  |
| Read or Write             | Write   | Read  | Read  |

Source: Nexla analysis, April 2018

# Conversion des données en Parquet (1)

- Chargez dans l'HDFS les csv des pays suivants : ca, fr, gb, us, de
- Créer le dossier suivant : `/user/raj_ops/youtube_parquet`, éditez ses droits en 777
- Exécutez le code du notebook import and transform
- Le dossier youtube\_parquet doit désormais contenir 5 dossiers

Le problème, c'est qu'ils appartiennent à l'utilisateur zeppelin et ne sont pas éditables par qui que ce soit d'autre (ce qui va nous emm\*\*\*\*\*er pour la suite)

# Conversion des données en Parquet (2)

Nous allons changer le propriétaire du dossier youtube\_parquet et les privilèges à l'aide de quelques commandes bash

- Se logger comme **root** au « Shell-in-a-box » et taper les trois commandes suivantes

```
su hdfs
```

```
hdfs dfs -chown -R raj_ops:hdfs /user/raj_ops
```

```
hdfs dfs -chmod -R 777 /user/raj_ops/youtube_parquet
```

raj\_ops est désormais propriétaire

# Création d'une **table interne**

Reprendre la requête précédemment générée lors de l'importation.

```
CREATE TABLE `youtube`.`videos` (  
  `video_id` STRING,  
  `trending_date` STRING,  
  .....  
) PARTITIONED BY (country STRING) # AJOUTEZ CECI  
STORED AS PARQUET
```

# Chargement des données

Exécutez cette commande pour chaque pays (remplacez `{pays}` par les deux lettres du pays que vous souhaitez charger)

```
LOAD DATA INPATH "/user/raj_ops/youtube_parquet/country=${pays}/data.parquet" INTO TABLE videos  
PARTITION(country= '${pays}')
```



# Allons voir le Warehouse

Sachant que nous venons de créer une **table interne**, cela implique que les données ont du être copiées dans un dossier géré par Hive

```
/warehouse/tablespace/managed/youtube.db/videos
```

# Requêtes sur notre nouvelle table

**NB** les partitions se comportent comme des colonnes (clause **WHERE**)

À l'aide d'une requête :

- Déénombrer les chaînes (channels) de chaque pays : `pays | nb(chaine)`
- Déénombrer les vidéos de chaque categories (pour chaque pays) : `pays | categories | nb(occurrence)`
- Trouvez les 10 vidéos les plus regardez par pays : `pays | title`

# Suppression de la table videos

Sans grande surprise :

**DROP TABLE** videos

Notez que cette opération efface les données du warehouse

Pour vous en assurer, allez voir

[/warehouse/tablespace/managed/youtube.db/](#)

# Création d'une **table externe**

Reprendre la requête précédemment utilisée et apportez les modifications suivantes

```
CREATE EXTERNAL TABLE `youtube`.`videos` (  
  `video_id` STRING,  
  `trending_date` STRING,  
  .....  
) PARTITIONED BY (country STRING)  
STORED AS PARQUET
```

# Création d'une **table externe**

Exécutez cette commande pour chaque pays (remplacez `${pays}` par les deux lettres du pays que vous souhaitez charger)

```
ALTER TABLE youtube.videos ADD PARTITION (country='${pays}')  
LOCATION '/user/raj_ops/youtube_parquet/country=${pays}'
```

# DROP PARTITION

Au besoin, vous pouvez supprimer une partition de la manière suivante

```
ALTER TABLE youtube.videos DROP PARTITION(country="$  
{country}")
```

# Vérification

Assurez-vous que votre table fonctionne à l'aide d'un **SELECT**

Remarquez que :

- Aucun dossier du nom de videos n'existe dans `/warehouse/tablespace/managed/youtube.db`
- En revanche, vous en trouverez un `/warehouse/tablespace/external/hive/youtube.db`

# Ajout de colonnes

Nous allons calculer un ratio vues/comment et stocker nos résultats dans une colonne du nom de « ratio »

```
ALTER TABLE videos ADD COLUMNS (ratio FLOAT);
```



# UPDATE (pas si simple)

Essayez d'update notre table afin d'ajouter le ratio pour chaque ligne

La syntaxe est similaire à celle de n'importe quel SGBD

```
UPDATE videos SET colonne=valeur WHERE condition;
```

# DELETE (pas si simple non plus)

Supprimer les lignes dont `video_error_or_removed` = `True`

```
DELETE from table  
WHERE condition ;
```

# Dans leurs termes

Hadoop is a batch processing system and Hadoop jobs tend to have **high latency** and incur substantial overheads in job submission and scheduling. As a result latency for Hive queries is generally very high (minutes) even when data sets involved are very small (say a few hundred megabytes). As a result it cannot be compared with systems such as Oracle where analyses are conducted on a significantly smaller amount of data but the analyses proceed much more iteratively with the response times between iterations being less than a few minutes. Hive aims to provide acceptable (but not optimal) latency for interactive data browsing, queries over small data sets or test queries.

**Hive is not designed for online transaction processing and does not offer real-time queries and row level updates.** It is best used for batch jobs over large sets of immutable data (like web logs).

# Morale de l'histoire

**Hive n'est pas un SGDB**, plus une abstraction sur des fichiers non-mutable (souvenez vous, HDFS, inaltérabilité) destinée à permettre un traitement global sous la forme de requêtes SQL

Le message fait toutefois mention d'une solution : des tables transactionnels

développons

# Conditions pour ACID

- Être une table **transactionnelle**
- Être une table stockée en **ORC**
- Être une table clusterisée

Une table clusterisée est :

- organisée en fonction du contenu d'une ou plusieurs colonnes.
- Le choix des colonnes influe sur les performances en cas de **GROUP BY** ou **WHERE** (// **INDEX BY**)

# Création de Tables transactionnelles | ACID

Nous allons supprimer notre table et la recréer en ajoutant

```
CREATE TABLE `youtube`.`videos` (  
  `video_id` STRING,  
  `trending_date` STRING,  
  .....  
  `country` STRING  
)  
CLUSTERED BY(category_id) INTO 3 BUCKETS  
STORED AS ORC  
TBLPROPERTIES ('transactional'='true');
```

# Problem solving

En tentant un LOAD DATA, vous allez vous rendre compte que vous aurez un problème

Quelle(s) solution(s) envisageriez vous ?

# Solution straightforward

Notre nouvelle table videos est une table transactionnelle. En conséquence, elle supporte **INSERT INTO**

- Créer une table temporaire (eg videosTemp) en Parquet (en tenant compte du partitionnement)
- **INSERT INTO** videos **SELECT** \* **FROM** videosTemp



# Retenons maintenant nos UPDATE / DELETE

Supprimez de la table l'ensemble des lignes  
`video_error_or_removed = True`

Créez une colonne **ratio\_comment** (Float) dont la valeur correspondra à **views / comment\_count**

# Passons à la vitesse supérieur : les jointures

Pour la suite de l'exercice, nous allons passer à une autre BDD du nom de **foodmart** (préexistante). Commençons simple

Trouvez le nom et le prénom des dix employés les mieux payés au 1997-01-01. Vous aurez besoin des tables suivantes :

- Salary
- Employee

# Passons à la vitesse supérieur : les jointures

Quel est le nom produit à avoir enregistré la plus grosse croissance de ventes (unit sales) entre 1997 et 1998. Vous aurez besoin

- sales\_fact\_1997
- sales\_fact\_1998
- product

Quelle région dispose du moins important ratio nb d'entrepôts / nombre de magasins ? Vous aurez besoin de

- Warehouse
- Store
- Region

# Création de tables agrégées (Scénario)

Maintenant que nous avons vu comment joindre et agréger, nous allons constituer quelque(s) table(s) à destination des décideurs. Ces derniers veulent évaluer la probabilité d'un défaut d'approvisionnement dans chaque région.

Vous allez devoir

- 1) Créer une table transactionnelle du nom de **agg\_region\_supply**
- 2) La peupler avec les données pertinentes
- 3) Mettre en place une visualisation pertinente de ces données agrégées

# Création de tables agrégées (Scénario)

La table devra contenir les données suivantes :

- Région\_id **Int**
- Région\_name **String**
- La présence d'un entrepôt de type (« Large Owned » cat 6) **Boolean**
- Le ratio entrepôt indépendant / entrepôt owned (pondéré par la taille de l'entrepôt) **Float**
- Le nombre moyen de stores fournis par les entrepôts indépendants **Float**
- Le nombre moyen de stores fournis par les entrepôts owned **Float**

# Mise en place de la visualisation

Vous allez devoir charger la table créé sous la forme d'un data frame spark que vous convertirez par la suite en df panda

Étape 1 : donner à Spark l'accès à toutes les BDD de HIVE

Il suffit, en tant que root, d'exécuter la commande suivante :

```
cp /etc/hive/conf/hive-site.xml /etc/spark2/conf
```

répondre **yes**

# Mise en place de la visualisation

Étape 2 : importer la table sous forme d'un dataframe

en **pyspark**, sous Zeppelin, il suffira de dire :

```
df = spark.sql("SELECT * FROM foodmart.$  
{nom_de_votre_table}")
```

# Optimisation : Indexation ?

Il est généralement recommandé d'indexer les champs les champs impliqués dans les clauses **WHERE** , **ORDER BY**

Cependant, cela n'est

pas recommandé

impossible depuis Hive 3.0

ni nécessaire :

Les format colonnaires (parquet, ORC) dispensent Hive de lire l'intégralité des lignes (à la différence des pointeurs de SGDB traditionnels)



# HIVE spécificités

## Support des **subqueries**

Hive accepte les subqueries mais exclusivement :

- Celles mentionnées dans un **FROM**
- À la condition d'utiliser un **alias**

```
SELECT total FROM  
(SELECT c1+c2 as total from  
my_table) my_query ;
```

# Hive spécificités

## Table externes

Données ne sont pas déplacées dans le {warehouse directory}

À la suppression de la table, seuls les métadonnées sont supprimées

## Tables internes

Données déplacées dans le {warehouse directory}

À la suppression de la table, métadonnées et fichiers sont supprimées