

## Atelier Pratique 1 : Data Warehouse Distribué avec Apache Hive

### Objectifs pédagogiques

À l'issue de cet atelier, les participants seront capables de :

1. Comprendre les concepts de Data Warehousing dans un contexte Big Data.
2. Concevoir un entrepôt de données en Hive (modèle en étoile ou flocon).
3. Créer et gérer des tables dimensionnelles et de faits.
4. Exécuter des requêtes OLAP sur des données volumineuses.
5. Optimiser les performances avec des formats de stockage (Parquet, ORC), partitionnement et bucketing.
6. Comprendre l'architecture distribuée d'Hive avec Hadoop.

### Prérequis techniques

- Cluster Hadoop 3.3.6 (Namenode, Datanodes, YARN)
- Hive avec PostgreSQL comme Metastore (Docker)
- Beeline ou CLI Hive
- SQL de base

### Partie 1 – Mise en place de l'environnement (Docker)

Tu disposez déjà d'un fichier `docker-compose.yml` avec les services suivants :

- `postgres` (metastore)
- `hive-server`
- `hive-metastore`

```
docker-compose up -d
```

Se connecter à Hive :

```
docker exec -u root-it hiveserver2 bash  
beeline -u jdbc:hive2://localhost:10000
```

## Partie 2 – Conception d'un entrepôt de données (schéma en étoile)

### Tables dimensionnelles

```
CREATE TABLE dim_client (  
    id_client INT,  
    nom STRING,  
    ville STRING,  
    pays STRING  
) ;  
  
CREATE TABLE dim_produit (  
    id_produit INT,  
    nom_produit STRING,  
    categorie STRING  
) ;  
  
CREATE TABLE dim_temps (  
    id_temps INT,  
    jour INT,  
    mois STRING,  
    annee INT  
) ;
```

### Table de faits

```
CREATE TABLE ventes (  
    id_vente INT,  
    id_client INT,  
    id_produit INT,  
    id_temps INT,  
    montant FLOAT  
)  
STORED AS PARQUET;
```

## Partie 3 – Chargement des données dans Hive

### Insérer les données dans les tables de dimensions

- Données pour **ventes**:

```
INSERT INTO ventes (id_vente,id_client,id_produit,id_temps,montant)
VALUES
(1,1,101,202301,150.50),
(2,2,102,202301,200.00),
(3,3,103,202301,120.75),
(4,1,104,202302,180.25),
(5,4,101,202303,220.30),
(6,5,102,202303,210.90),
(7,2,103,202304,185.60),
(8,3,104,202304,160.80),
(9,1,101,202305,145.30),
(10,4,102,202305,230.00);
```

- Données pour **dim\_client**:

```
INSERT INTO dim_client VALUES
(1, 'Ali', 'Casablanca', 'Maroc'),
(2, 'Sanae', 'Tanger', 'Maroc'),
(3, 'Youssef', 'Fès', 'Maroc'),
(4, 'Fatima', 'Rabat', 'Maroc'),
(5, 'Mohamed', 'Agadir', 'Maroc');
```

- Données pour **dim\_produit**:

```
INSERT INTO dim_produit VALUES
(101, 'Laptop Lenovo', 'Informatique'),
(102, 'Smartphone Samsung', 'Téléphonie'),
(103, 'Clavier Logitech', 'Accessoires'),
(104, 'Écran Dell', 'Informatique');
```

- Données pour **dim\_temps**:

```
INSERT INTO dim_temps VALUES
(202301, 1, 'Janvier', 2023),
(202302, 5, 'Février', 2023),
(202303, 12, 'Mars', 2023),
(202304, 20, 'Avril', 2023),
(202305, 28, 'Mai', 2023);
```

## Partie 4 – Requêtes OLAP

### -- CA total par année

```
SELECT t.année, SUM(v.montant)
FROM ventes v
JOIN dim_temps t ON v.id_temps = t.id_temps
GROUP BY t.année;
```

### -- Top 3 produits

```
SELECT p.nom_produit, SUM(v.montant) AS total
```

```
FROM ventes v
JOIN dim_produit p ON v.id_produit = p.id_produit
GROUP BY p.nom_produit
ORDER BY total DESC LIMIT 3;

-- Répartition géographique
SELECT c.pays, COUNT(*)
FROM ventes v
JOIN dim_client c ON v.id_client = c.id_client
GROUP BY c.pays;
```

# Partie 5 : Visualisation des requêtes Hive avec Superset – Étapes détaillées

## Objectif

Permettre aux étudiants d'exécuter des requêtes OLAP sur Hive et de visualiser les résultats à l'aide d'un tableau de bord interactif avec **Apache Superset**.

## Étape 1 – Ajouter le service Superset dans Docker

Dans ton fichier `docker-compose.yml`, ajoute le service suivant :

```
superset:
  image: apache/superset
  container_name: superset
  restart: unless-stopped
  ports:
    - "8088:8088"
  environment:
    SUPERSET_SECRET_KEY: 'thisISaSECRET_key'
    ADMIN_USERNAME: admin
    ADMIN_PASSWORD: admin
    ADMIN_FIRST_NAME: Admin
    ADMIN_LAST_NAME: User
    ADMIN_EMAIL: admin@example.com
  volumes:
    - superset_home:/app/superset_home
```

```
depends_on:  
  - hiveserver2  
networks:  
  - hive
```

Ajoute aussi dans la section `volumes` (à la fin du fichier) :

```
volumes:  
  hive-db:  
  warehouse:  
  superset_home:
```

Puis exécute :

```
docker-compose up -d
```

## Étape 2 – Installer les outils nécessaires dans le conteneur Superset

### 1. Ouvrir un terminal root dans le conteneur Superset :

```
docker exec -u root -it superset bash
```

### 2. Installer les outils de compilation nécessaires pour les paquets Python natifs :

```
apt-get update && apt-get install -y build-essential g++
```

### 3. Installer les bibliothèques Python permettant à Superset de se connecter à Hive :

```
pip install "pyhive[hive]" thrift sasl thrift-sasl
```

Cette commande installe :

- Le driver PyHive compatible avec SQLAlchemy
- Le protocole Thrift
- Le support SASL (authentification HiveServer2)

Ensuite, tape `exit` pour sortir du conteneur.

## Étape 3 – Démarrer Superset et accéder à l'interface

Dans le navigateur, ouvre l'URL suivante :

**Identifiants par défaut :**

- **Nom d'utilisateur :** admin
- **Mot de passe :** admin

## Étape 4 – Connecter Superset à Hive

1. Dans Superset, aller dans le menu **Data > Databases**
2. Cliquer sur **+ Database**
3. Choisir "**Connect this database using a SQLAlchemy URI string**"
4. Dans le champ **SQLAlchemy URI**, entrer :

```
hive://hive@hiveserver2:10000/default
```

`hiveserver2` est le nom du conteneur Hive défini dans `docker-compose.yml`.

5. Cliquer sur **Test Connection**
6. Si tout est correct, le message `It looks good!` s'affiche
7. Cliquer sur **Save**

## Étape 5 – Créer une requête Hive

1. Aller dans **SQL Lab > SQL Editor**
2. Sélectionner la base de données Hive et le schéma default
3. Exemple de requête OLAP :

```
SELECT t.annee, SUM(v.montant) AS total  
FROM ventes v  
JOIN dim_temps t ON v.id_temps = t.id_temps  
GROUP BY t.annee
```

4. Cliquer sur **Run** pour exécuter la requête
5. Cliquer sur **Explore** pour visualiser les résultats

## Étape 6 – Créer un tableau de bord

1. Aller dans **Dashboards > + Dashboard**

2. Donner un nom (ex. : "Analyse des ventes Hive")
3. Ajouter les graphiques précédemment créés depuis SQL Lab
4. Organiser et personnaliser le tableau de bord (barres, camemberts, etc.)
5. Cliquer sur **Save**

## Partie 6 – Optimisations avancées

### Partitionnement

```
CREATE TABLE ventes_part (
    id_vente INT,
    id_client INT,
    id_produit INT,
    montant FLOAT
)
PARTITIONED BY (année INT)
STORED AS PARQUET;
```

### Bucketing

```
CREATE TABLE ventes_bucketed (
    id_vente INT,
    id_client INT,
    id_produit INT,
    montant FLOAT
)
CLUSTERED BY (id_client) INTO 4 BUCKETS
STORED AS ORC;
```

Activer :

```
SET hive.enforce.bucketing = true;
SET hive.enforce.sorting = true;
```

## Partie 6 – Travaux Pratiques

1. Ajouter une nouvelle dimension `magasin`.
2. Créer une vue matérialisée `CA_par_pays_et_année`.

## Big Data : Architectures de stockage | 2025

3. Tester les performances avec et sans partitionnement.
4. Comparer les formats : TEXTFILE, PARQUET, ORC.