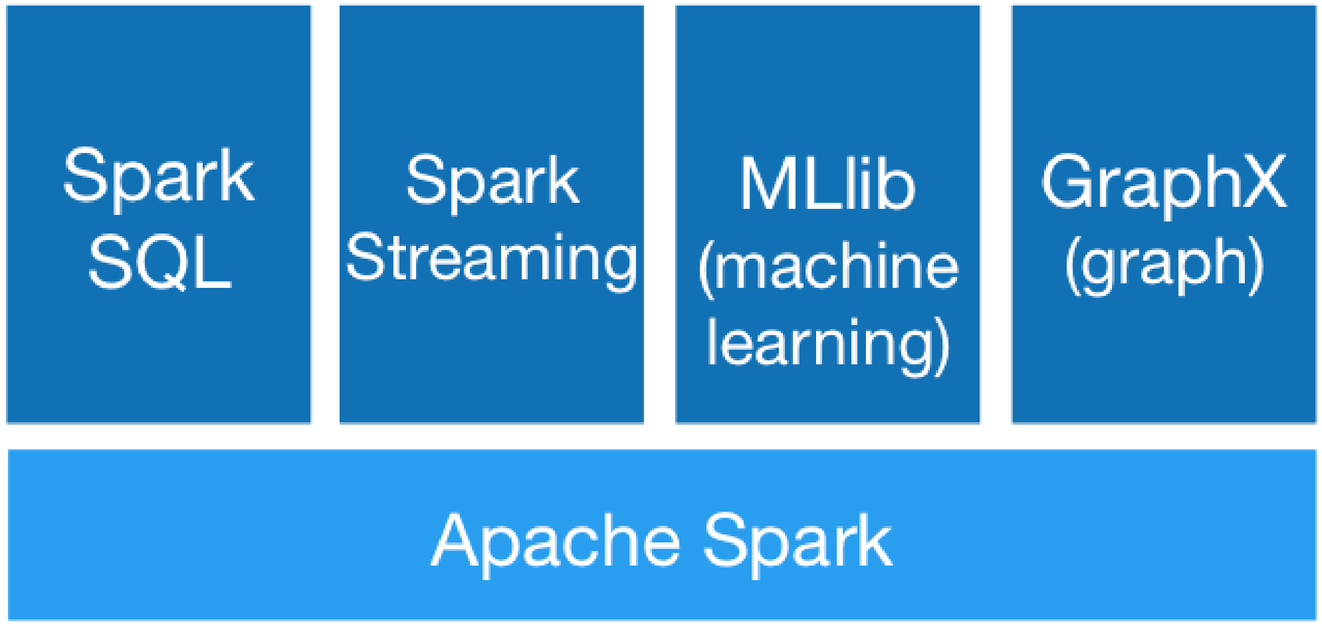
Big Data Analytics

« Pilotage de la performance pour une bonne gouvernance des entreprises »

CHAPITRE 3 – Spark SQL

Spark Stack



- ***Spark SQL :*** pour le traitement de données (SQL et non structuré)
- ***Spark Streaming :*** traitement de flux de données en direct (live streaming)
- ***MLlib :*** Algorithmes Machine Learning
- ***GraphX :*** Traitement de graphes

Qu’est ce que Spark SQL ?

- Spark permet de manipuler d’importants volumes de données en utilisant une API de bas niveau.
- Spark SQL, composant du Framework Spark, est utilisé pour effectuer des traitements sur des données structurées en exécutant des requêtes de type SQL sur les données Spark.
- Spark SQL permet d’exécuter des requêtes ad-hoc après une étape d’ETL sur des données stockées sous différents formats (JSON, des données stockées dans des bases SQL Server, mySQL, Oracle, …)
- Pour simplifier l’exploration des données, Spark SQL offre une API de plus haut niveau avec une syntaxe SQL.

Qu’est ce que Spark SQL ?

- Spark SQL permet de réaliser de nombreuses opérations rapidement sans écrire de code.

- Contrairement aux RDDs, les interfaces fournies par Spark SQL fournissent davantage d'informations sur la structure des données et sur le calcul en cours.

- Spark SQL utilise ces informations supplémentaires pour effectuer des optimisations.

Datasets & Dataframes

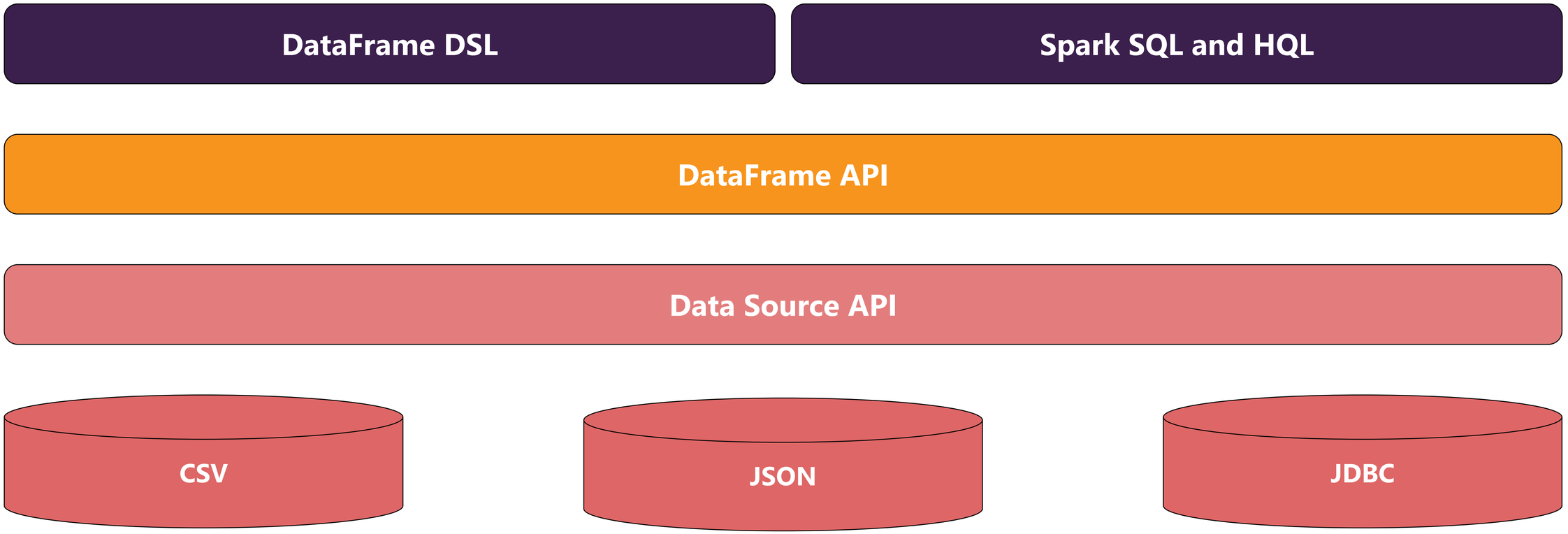
- Un Dataset est une collection distribuée de données.
- Un Dataset offre les avantages des RDD (typage puissant, capacité à utiliser de puissantes fonctions) avec les avantages du moteur d'exécution optimisé de Spark SQL.
- Un Dataset peut être construit à partir d'objets JVM, puis manipulé à l'aide de transformations fonctionnelles (map, filter, ...).

- Un DataFrame est un Dataset organisé en colonnes nommées.
- Un DataFrame est conceptuellement équivalent à une table dans une base de données relationnelle mais avec des optimisations plus riches.
- Les DataFrames peuvent être construits à partir plusieurs sources (des fichiers de données structurés, des tables Hive, des BDs externes ou des RDD existants.)

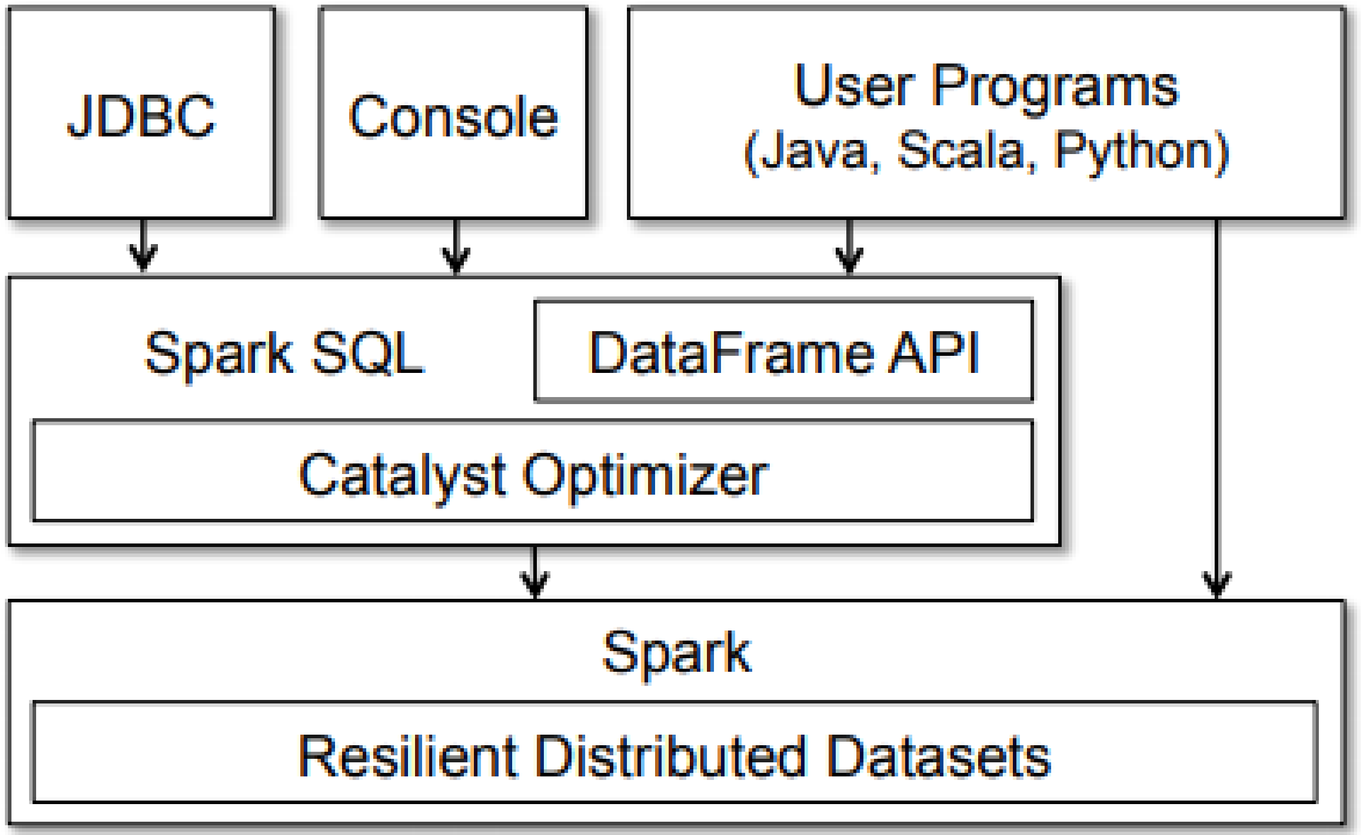
SQL

- Avec SQL, plusieurs choses nous viennent à l’esprit :
  - Base de données
  - Schéma et types de données
  - Tables / vues
  - Fonctions SQL
  - Fonctions définies par l'utilisateur
  - Métadonnées
  - Plans d'exécution et optimisation
  - Clients SQL et connectivité
  - Constructions DDL et DML

Architecture de Spark SQL



Interfaces Spark SQL et interaction avec Spark



[Spark SQL: Relational Data Processing in Spark; Databricks Inc., MIT CSAIL, AMPLab, UC Berkeley]

Tables dans Spark SQL

- Spark SQL prend en charge deux types de tables.
  - Tables gérées (« *Managed Tables* »),
  - Tables non gérées ou tables externes (« *Unmanaged Tables* »).

- Managed Tables :
  - Spark stocke une table gérée dans l'emplacement du répertoire de la base de données.
  - Si on supprime une table gérée, Spark supprimera le fichier de données ainsi que le sous-répertoire de la table.

Tables dans Spark SQL

- Spark SQL prend en charge deux types de tables.
  - Tables gérées (« *Managed Tables* »),
  - Tables non gérées ou tables externes (« *Unmanaged Tables* »).

- Unmanaged Tables :
  - Les fichiers non gérés sont des tables externes. Ils résident quelque part en dehors du répertoire de la base de données.
  - Si vous supprimez une table non gérée, Spark supprimera l'entrée de métadonnées pour cette table et, en raison de cette instruction de suppression, ne pourrez pas accéder à cette table à l'aide de Spark SQL.
  - Le fichier de données pour cette table non gérée réside toujours à l'emplacement d'origine.

Pourquoi avons-nous besoin de tables externes ?

Besoin de « *Unmanaged Table* »

- Supposons que certaines données se trouvent dans un autre emplacement du système de fichiers ou dans un autre système de stockage (une base de données JDBC, MongoDB, …),
- Ces données sont stockées, maintenues et gérées par un autre système. Spark ne le possède pas,
- On souhaite les rendre disponible pour une application de base de données Spark et ses utilisateurs de la même manière que les tables gérées,
- On n'a pas besoin de faire une copie on se réfère à une table gérée localement.