MU5IN852
Bases de Données Large Echelle

octobre 2020

hubert.naacke@lip6.fr

Plan des séances BDLE

- Données multidimensionnelles, OLAP
- Map reduce
  - Spark
  - Algo de ML en map reduce
- Cluster computing
  - exécution distribuée et optimisation de requêtes
- Big graphs
  - Bulk synchronous parallel processing
  - Modèle et langage de requêtes pour des grands graphes

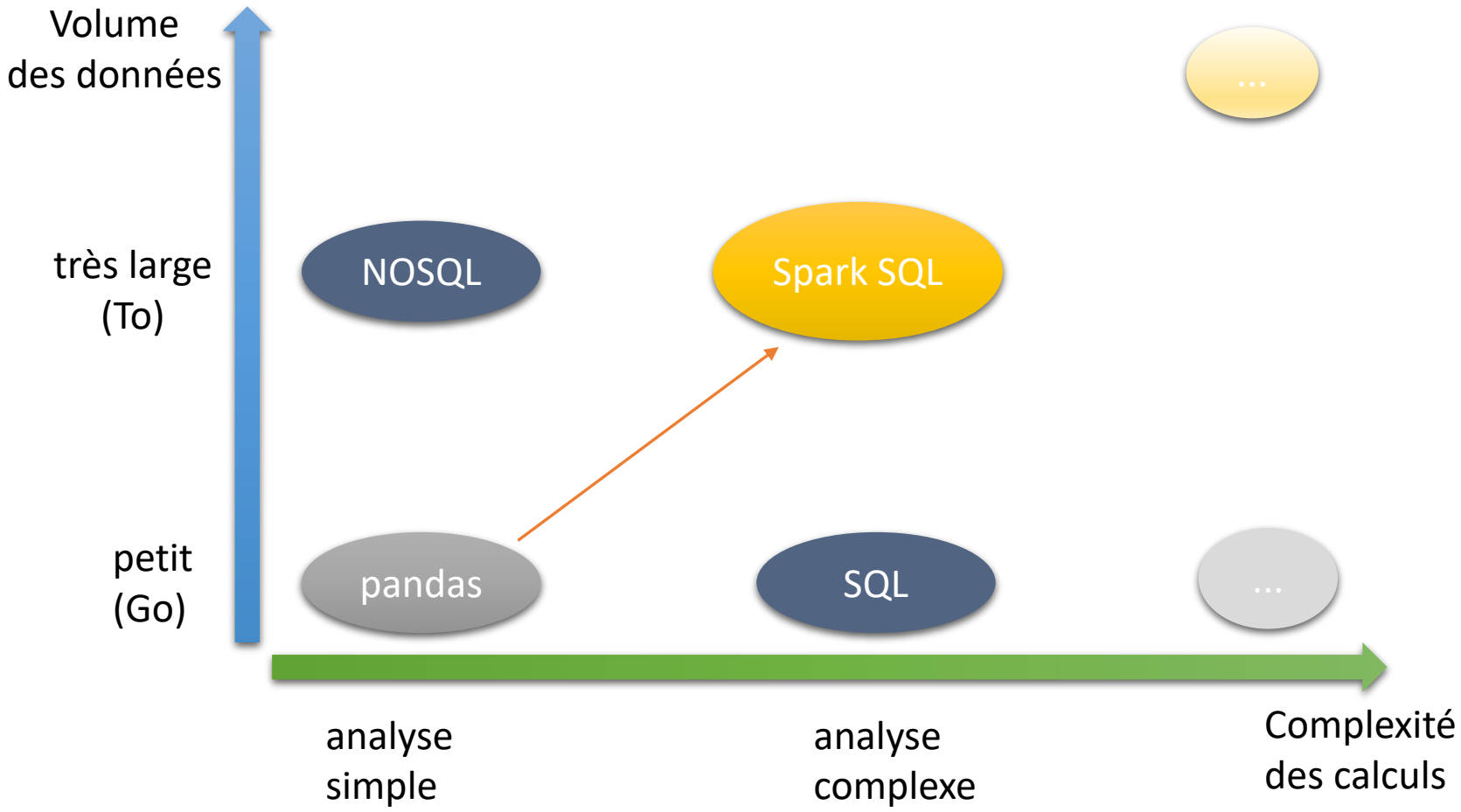
Transversal

- savoir faire sur Spark et la gestion de big data

SQL à large échelle

septembre 2020

Défis


Volume
des données
très large
(To)
petit
(Go)
NOSQL
Spark SQL
...
pandas
SQL
...
analyse
simple
analyse
complexe
Complexité
des calculs

Motivations et Objectifs

- Volume : analyser un fichier quelle que soit sa taille
  - S'affranchir de la limite imposée par la capacité d'une machine
  - manipuler efficacement un fichier qui ne tient pas dans la mémoire d'une machine
- Diversité des données : analyser des fichiers «bruts»
  - Données peu structurées, incomplètes
- Poser une requête **déclarative** sur un fichier
  - Moteur SQL pouvant lire directement et efficacement des fichiers
- Analyse avancée
  - Disposer de nombreuses fonctions d'analyse
- Langage de requête extensible
  - SQL intégrant des fonctions définies par l'utilisateur : UDF

Modèle de données

- Un fichier est représenté par une TABLE relationnelle
  - Schéma d'une table
    - NomTable(attribut 1, …., attribut n)
  - Un attribut a un type
    - Numérique : Int, long
    - Chaîne de caractères
    - Type tableau : un attribut a une liste de valeurs

Modèle de stockage

- Stockage adapté à la conception de pipeline
  - Exple: préparation de données
- Seuls les fichiers sont stockés
- Une table n'est pas stockée
- Possibilité de sauvegarder le résultat d'une requête dans un fichier
- Persistence du stockage
  - Stockage permanent sur disque
  - Stockage *volatile* en mémoire

Modèle d'exécution

- Une table est une vue virtuelle sur un fichier
- Une requête est une vue virtuelle sur des tables et/ou des requêtes
- Une requête est **définie** sans être exécutée
  - apporte de la souplesse pour définir une requête en réutilisant des requêtes précédemment définies
- Exécution à la demande
  - Afficher le résultat d'une requête
  - Matérialiser ou stocker le résultat d'un calcul intermédiaire long pour le réutiliser plus rapidement

Solution qui passe à l'échelle

- Architecture big data orientée service
  - Système offrant un service de requêtes
  - Système distribué
    - capacité non limitée
    - haute disponibilité
- Transparence
  - Fonctionne de manière identique avec 1 ou N machines
  - Permet de séparer :
    - La conception d'un scénario d'analyse sur un laptop
    - La mise en production sur un cluster

Exemples de cas d'usage

- Analyse de traces utilisateurs
  - Fichier: log
- Analyse de réseaux sociaux
  - Flux
- Préparation de données avant tâches de mining, ML, …

SQL avec l'interface pyspark

- pyspark : interface pour connecter une appli python avec le système spark
- Avantages
  - Langage familier
  - Intégration facilitée avec les lib python fréquemment utilisées
- Inconvénient
  - Langage différent de scala utilisé nativement
    - Inconvénient mineur quand python invoque des opérations natives en scala

→ Connaitre la syntaxe pyspark

Workflow en SQL : entrée-sortie

- Lire un fichier
  - F = spark.read.csv("fichier.csv")
  - F = spark.read.json("fichier.json")
- Ecrire le résultat d'une requête *req*
  - *req*.write.mode("overwrite").option("header", "true").csv(fichier)
  - *req*.write.mode("overwrite").json(fichier)
- Charger en mémoire, dans le système spark, le résultat d'une requête :
  - *req*.persist()
  - *req*.count()
- Afficher les n premiers tuples
  - *req*.show(n, False)
- Décrire les valeurs des attributs
  - *req*.describe("att1", "att2", ...).show()
  - Calcule pour chaque attribut : min, max, avg, sdtdev, count
- Schéma : structure d'une table
  - *req*.printSchema()

Workflow en SQL : Requête

- Requête écrite en SQL
  - F.createOrReplaceTempView("F")
  - req = spark.sql(‘’’ SELECT ...
    FROM F
    WHERE... GROUP BY… ‘’’)
- Résultat final d’une analyse
  - resFinal = req.collect()
  - seulement si req est « petit »
    - Le résultat de la requête *req* est calculé dans le système spark
    - Le tableau resFinal est alloué dans le noyau python

Workflow en SQL : pipeline

- Deux méthodes pour enchainer des requêtes SQL
- DAG de requêtes
  - r2 = spark.sql(‘‘‘select … from R1 where … ‘‘‘)
    .createOrReplaceTempView("R2")
  - r3 = spark.sql(‘‘‘ select … from R1, R2 where … ‘‘‘)
    .createOrReplaceTempView("R3")
- Requête nommée en SQL avec le mot clé **WITH**
  - r3 = spark.sql(‘‘‘ WITH R2 as (select … from R1 ….)
    select * from R2 …
    ‘‘‘)

Fonctions SQL prédéfinies

- Fonctions mathématiques et statistiques
- Manipulation des dates
  - date_format, date_add, date_part, …

Voir les fonctions SQL fournies par spark dans

spark.apache.org/docs/latest/api/**sql**

Fonctions : traitement des chaines de caractères

- Découper une chaine
  - split(chaine, séparateur)
  - Le séparateur peut être une expression régulière regex
    - Exple de séparateur : "[^a-z0-9]+" ou "\\W+"
- Extraction de motif
  - regexp_extract( )
  - exple : extraire un nombre entier placé entre parenthèses dans une chaine
    - cast(regexp_extract(att1, '**(\\\d+)**') as int) as durée
  - avec remplacement
    - regexp_replace(chaine, regex, rempl)
- Changement de casse
  - lower( ), upper( ) conversion en minuscules ou majuscules
- Nettoyer les espaces
  - trim, rtrim, ltrim
- Concaténation
  - concat(a1, …), avec séparateur: concat_ws(séparateur, chaine1, …)

Fonctions : manipulation de structures imbriquées

- Nest
  - imbrication par agrégation : group by
    - collect_list(attr) : génère un tableau contenant les valeurs d'attr pour chaque groupe.
    - collect_list((attr1, attr2)) : génère un tableau de tuples
  - Manipulations de tableaux
    - element_at(tab, position), find_in_set( )
    - algèbre: array_intersect, …
    - flatten d'un tableau de tableau
  - Quantification
    - every, exists
- Unnest
  - Désimbrication avec explode(tableau)
    - génère un tuple pour chaque valeur du tableau

Fonctions UDF

UDF: User Defined Function

Appliquer une fonction sur chaque tuple d'une table

- Définir une fonction python

```
def maFonction(param1, ...):
    result = ....
    return result
```

- Déclarer une fonction

```
spark.udf.register("maFonction", maFonction, IntegerType())
```

- Invoquer une fonction en SQL
  - requête SQL sur la table T(a, b, ...)

```
SELECT a, maFonction(b) as nouveauB
FROM T
```

Types SQL représentés en python

- Voir la doc pyspark.sql.types

  https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#module-pyspark.sql.types

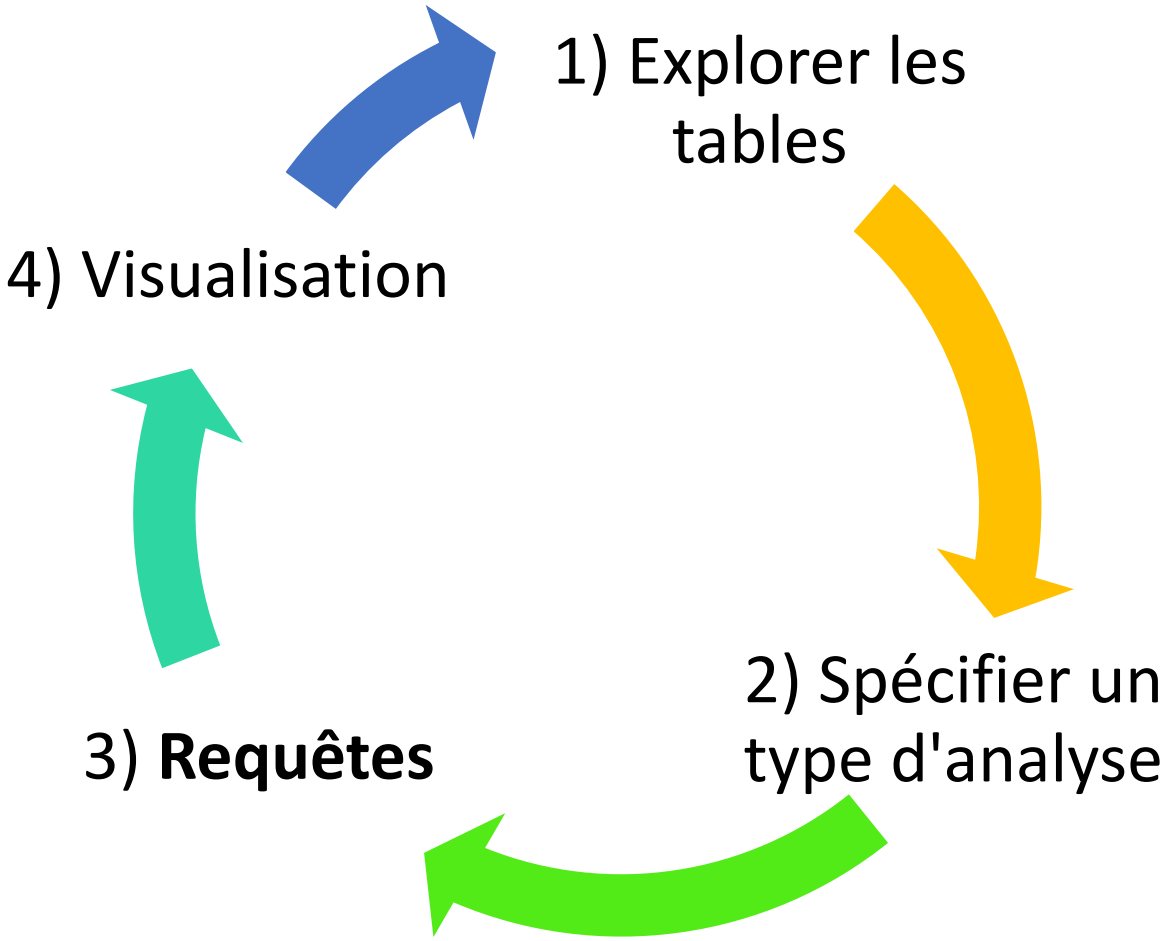
- Nombres et chaines de caractères
  - IntegerType(), LongType(), DoubleType(), StringType()
- Collections
  - ArrayType(type), MapType(typeClé, typeValeur)...
- Tuple
  - StructType([StructField(nom, type, nullable), ...])

Projet 1 : IMDB

- Explorer des fichiers de données avec pyspark

But du projet BDLE

Objectifs des séances



Savoir faire : outils

shell	Cloud	notebook	pyspark
• grands fichiers	• import/export • partage	• Jupyter • Colab • Databricks	• SQL avancé • viz mplotlib, …

Savoir faire : autonomie en TP

Détecter

- Comportement imprévu
- Résultat incorrect

Isoler

- Etat **avant** le problème
- Etat simplifié

Résoudre

- Reproduire
- Comprendre

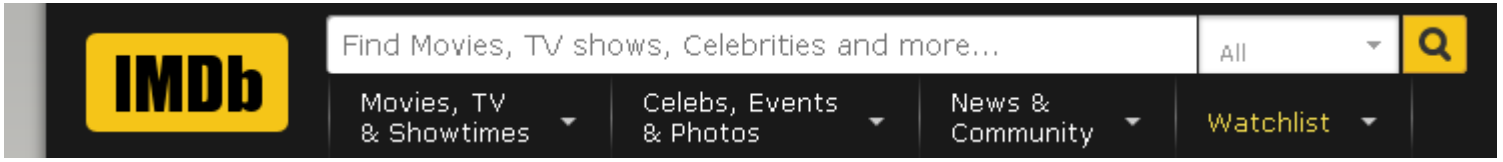


Schéma de la base imdb

- Les personnes : Name, Person_Info,
- Les films : Title, Movie_Info
- Les sociétés de production : Company_name

- Les associations :
  - film/personne : Cast_Info
  - film/société : Movie_companies

Schéma d'une personne

- Une personne : Name, Person_Info
- Casting : une personne a un ou plusieurs rôles dans un film

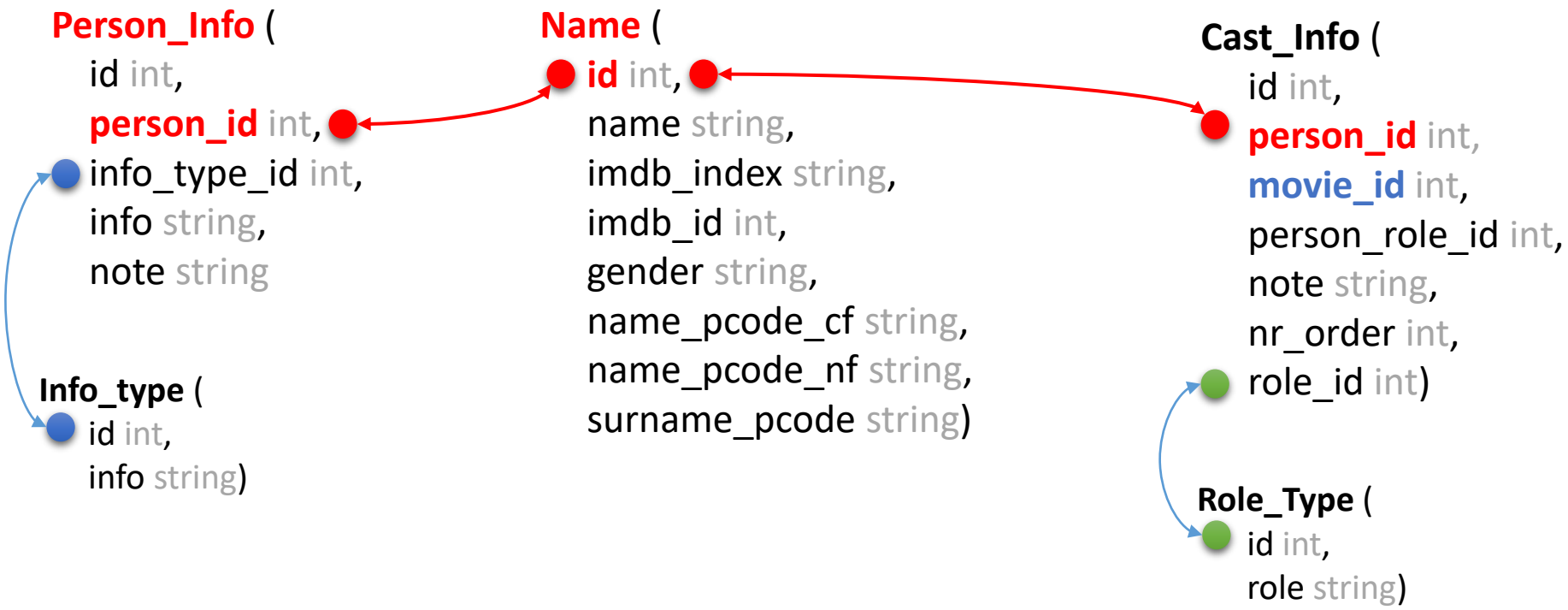


Schéma d'un film

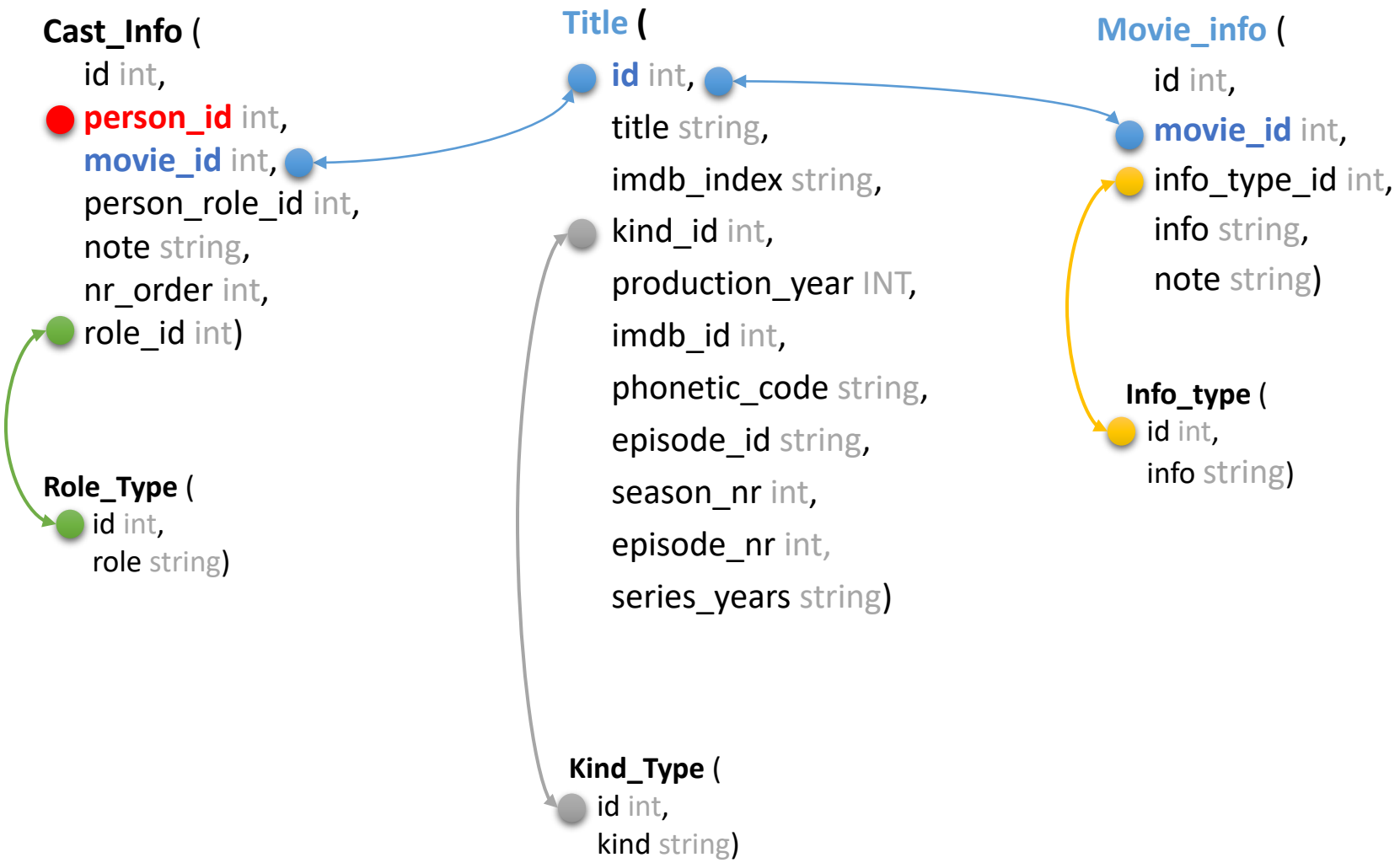
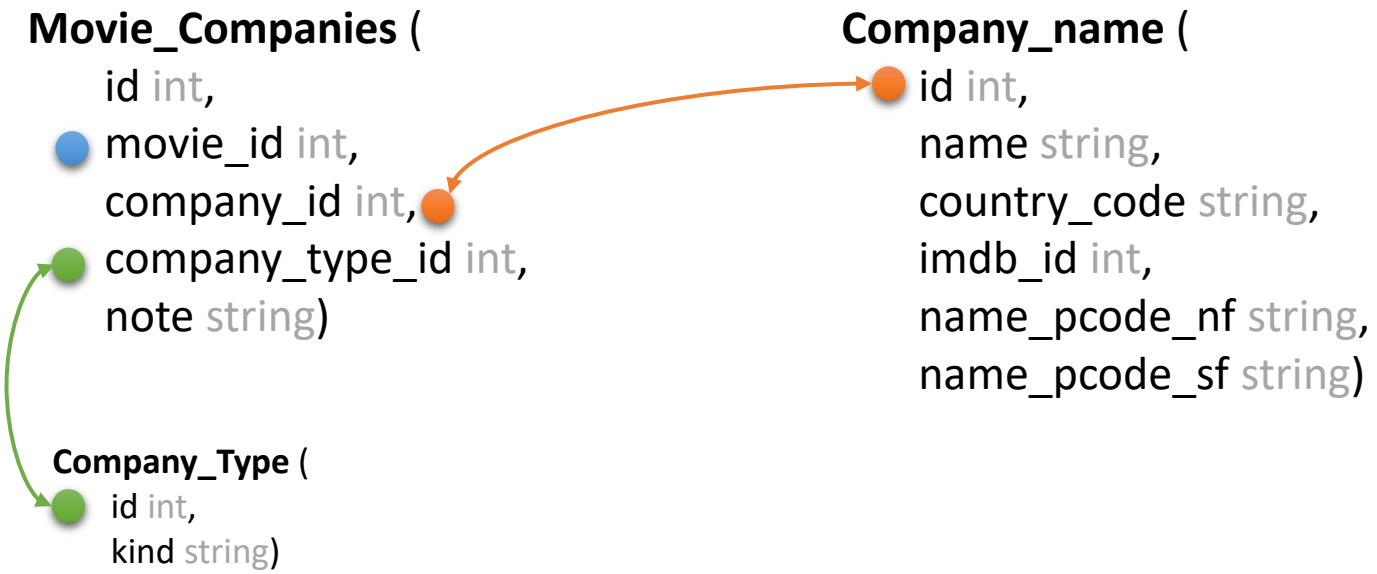


Schéma d'une société de production



Datasets

- Dossier partagé : PUBLIC_DATASET
  https://nuage.lip6.fr/s/H3bpyRGgnCq2NR4
- Fichiers IMDB dans le dossier
  - imbd/vldb2015/csvfiles_sample001

- Contient aussi d'autres datasets à disposition

Travaux pratiques

Outil scalable

- Plateforme Spark
  - Moteur SQL
    - Databricks community : community.cloud.databricks.com
  - Interface : Notebook mixant SQL et python
    - Supporte la syntaxe SQL : tag %sql
    - Le résultat d'une requête SQL est manipulable en python

Travaux pratiques avec Databricks

- Lire les indications sur la page de l'UE BDLE
  http://www-bd.lip6.fr/wiki/site/enseignement/master/bdle/tmes/databricks
- Importer les datasets dans Databricks
- Importer un notebook avec des exemples
- Obtenir des ressources de calcul
  - Démarrer un cluster
  - Associer un notebook à un cluster
- Se familiariser avec l'interface utilisateur du notebook

Biblio

- **Spark SQL : Relational Data Processing in Spark**
  - **in SIGMOD 2015**
  - Databricks, MIT CSAIL, UC Berkeley AmpLab

    https://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf

    https://dl.acm.org/doi/pdf/10.1145/2723372.2742797
- Site de l'équipe BD
  - https://www-bd.lip6.fr/wiki/site/enseignement/start
  - Requêtes avancées sur données semi-structurées
    - UE Modèles et langages pour les BD avancées
      - M1 4IN801