Bases de données documentaires et distribuées
Cours NFE04
Le langage Pig latin

Auteur : Philippe Rigaux

Département d'informatique
Conservatoire National des Arts & Métiers, Paris, France

Pig Latin

Motivation : définir un langage de haut niveau, type SQL, qui s'appuie sur un moteur d'exécution MapReduce.

Une "requête" Pig définit un opérateur qui prend une **relation** en entrée et produit une autre relation en sortie.

Les requêtes consistent en une **séquence** d'instructions suivant généralement le schéma suivant :

1. L'instruction **LOAD** charge les données d'un fichier (HDFS) sous la forme d'une **relation** (liste de n-uplets).
2. Une série de transformations traite les données.
3. L'instruction (optionnelle) **STORE** écrit le résultat dans le système de fichiers (HDFS) ; on peut aussi faire un **DUMP** pour afficher à l'écran.

Les requêtes sont evaluées par une **combinaison** de *jobs* MapReduce.

Tester Pig

Pig est un programme Java qui s'exécute en ligne de commande.

- Télécharger depuis `http://pig.apache.org`.
- Décompresser dans un répertoire, disons `pighome`
- Placer le répertoire `pighome/bin` dans le chemin d'exécuton (`PATH`)
- Deux modes d'exécution

  local on lit les données sur le disque local, on évalue sans MapReduce

  MapReduce on exécute dans un *cluster* MapReduce, en lisant depuis HDFS.

Pour exécuter en local :

```
pig -x local
grunt>run <script>.pig
```

Exemple de données en entrées (format tabulaire)

Un fichier texte, un n-uplet par ligne, champs séparés par des tabulations.

Fichier : `publishers.txt`

```
Fundations of Databases Addison-Wesley USA
Fundations of Databases Vuibert France
Web Data Management and Distribution CUP GB
```

Fichier : `books.txt`

```
1995 Fundations of Databases Abiteboul
1995 Fundations of Databases Hull
1995 Fundations of Databases Vianu
2010 Web Data Management Abiteboul
2010 Web Data Management Manolescu
2010 Web Data Management Rigaux
2010 Web Data Management Rousset
2010 Web Data Management Senellart
```

En pratique

On peut écrire une fonction Java qui charge les données des fichiers et les transforme en relation.

Premier exemple

On regroupe les auteurs par livre.

```
-- Chargement
books = load 'books.txt'
    as (year: int, title: chararray, author: chararray) ;

-- Regroupement
group_auth = group books by title;

-- Restruturation
authors = foreach group_auth generate group, books.author;

-- Affichage
dump authors;
```

Exécution

```
pig -x local
grunt>run groupby.pig
```

Modèle de données

Modèle semi-structuré, type XML/JSON. Imbrication de structures ensemblistes ("bags") et de n-uplets.

Exemple : le bag `group_auth`.

```
(
  Fundations of Databases,
  { (1995,Fundations of Databases,Abiteboul),
    (1995,Fundations of Databases,Hull),
    (1995,Fundations of Databases,Vianu)
  }
)
```

Imbrication sans limite, mais pas de références ou dépendance externe : un "document" autonome. **Objectif parallélisation** !

Les champs peuvent être référencés par leur nom (quand il y a un schéma) ou par leur position (ex, `$1`).

Un exemple complet : le fichier

Fichier : `journal-small.txt`

```
2005    VLDB J. Model-based approximate querying in sensor networks.
1997    VLDB J. Dictionary-Based Order-Preserving String Compression.
2003 SIGMOD Record Time management for new faculty.
2001    VLDB J. E-Services - Guest editorial.
    2003 SIGMOD Record Exposing undergraduate students to system internals.
    1998    VLDB J. Integrating Reliable Memory in Databases.
    1996    VLDB J. Query Processing and Optimization in Oracle Rdb
    1996    VLDB J. A Complete Temporal Relational Algebra.
    1994 SIGMOD Record Data Modelling in the Large.
    2002 SIGMOD Record Data Mining: Concepts and Techniques - Book Review.
```

Un exemple complet : le programme

Regroupement des publications par année.


```
-- Chargement des documents de journal-small.txt
articles = load 'journal-small.txt'
    as (year: chararray, journal:chararray, title: chararray) ;
sr_articles = filter articles BY journal=='SIGMOD Record';
year_groups = group sr_articles by year;
count_by_year = foreach year_groups generate group,
      COUNT(sr_articles.title);
dump count_by_year;
```

À décomposer...

Principaux opérateurs Pig

Operator	Description
foreach	Applique une expression à chaque document
filter	Filtre les tuples sur certains critères
distinct	Elimination des doublons
join	Jointures de deux *bags*
group	Regroupement
cogroup	Association de n-uplets provenant de deux sources
cross	Produit cartésien de deux sources
order	Tri
limit	Limite la taille des données traitées
union	Union de deux sources (dont les schémas peuvent être très différents)
split	Partition d'une relation selon certains critères

Opérateur de "dégroupage"

`flatten` "déploie" une imbrication (`flatten.pig`).

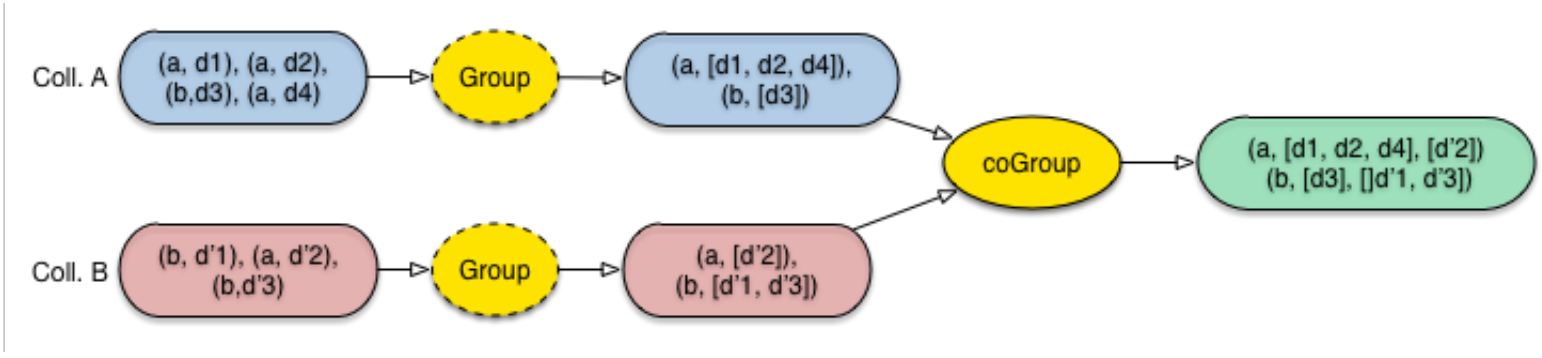
```
-- Take the 'authors' bag and flatten the nested set
flattened = foreach authors
  generate group, flatten($1);
```

Appliqué à la relation `authors` créée précédemment, on obtient :

```
(Foundations of Databases,Abiteboul)
(Foundations of Databases,Hull)
(Foundations of Databases,Vianu)
(Web Data Management,Abiteboul)
(Web Data Management,Manolescu)
(Web Data Management,Rigaux)
(Web Data Management,Rousset)
(Web Data Management,Senellart)
```

L'opérateur cogroup

Permet d'assembler des n-uplets sur un ensemble de critères.



Exemple (cogroup.pig) :

```
publishers = load 'webdam-publishers.txt'
    as (title: chararray, publisher: chararray) ;
cogrouped = cogroup flattened by group,
                    publishers by title;
```

Résultat (mis en forme)

Pour chaque valeur de regroupement, deux ensembles imbriqués, provenant de chacune des sources.

```
(Foundations of Databases,
  { (Foundations of Databases,Abiteboul),
    (Foundations of Databases,Hull),
    (Foundations of Databases,Vianu)
  },
  {(Foundations of Databases,Addison-Wesley),
   (Foundations of Databases,Vuibert)
  }
)
```

Une sorte de jointure, dans un état intermédiaire, permis par le modèle semim-structuré.

Jointures

Come précédemment, mais produit un résultat “mis à plat”.

On effectue le produit cartésien des n-uplets imbriqués (`join.pig`).

```
-- Take the 'flattened' bag, join with 'publishers'
joined = join flattened by group, publishers by title;
```

```
(Foundations of Databases,Abiteboul,
      Fundations of Databases,Addison-Wesley)
(Foundations of Databases,Abiteboul,
      Fundations of Databases,Vuibert)
(Foundations of Databases,Hull,
      Fundations of Databases,Addison-Wesley)
(Foundations of Databases,Hull,
     ...
```

Un exemple à tester

Avec ajout de filtres pour sélectionner certain n-uplets.

```
-- Chargement de books.txt
books = load 'books.txt'
    as (year: int, title: chararray, author: chararray) ;
-- On prend les livres de Victor Vianu
vianu = filter books by author == 'Vianu';
--- Chargement de publishers.txt
publishers = load 'publishers.txt'
    as (title: chararray, publisher: chararray) ;
-- Jointure sur le titre
joined = join vianu by title, publishers by title;
-- Groupement sur le nom de l'auteur
grouped = group joined by vianu::author;
-- Maintenant, comptons les editeurs (exercice: supprimer les
count = foreach grouped generate group,
           COUNT(joined.publisher);
```

À vous de jouer.