TP table de données, fusion

Traitement de données en tables

Première NSI Lycée du Parc

Table des matières

Crédits

Ce TP est largement inspiré du chapitres 18 du manuel NSI de la collection Tortue chez Ellipse, auteurs : Ballabonski, Conchon, Filliatre, N'Guyen ;

1 Réunion de tables

Point de cours 1

Si on dispose de deux **tables de données** avec la même structure (même nombre d'attributs avec noms et domaines de valeurs identiques), il est possible de les fusionner en construisant leur **réunion** dans une même table en concaténant simplement les lignes / enregistrements.

L'opérateur `+` de concaténation de tableaux permet de réaliser très simplement cette opération en Python. On donne ci-dessous un exemple de réunion de tables de notes de différentes classes à un devoir commun de NSI. Dans un second exemple, on observe que si les tables n'ont pas la même structure (attributs différents en noms et en nombre), on peut les concaténer mais on ne peut plus manipuler le résultat comme une table :

```
>>> premiere1 = [{'élève' : 'Donald', 'note' : 18},{'élève' : 'Ada', 'note
    ' : 20}]
>>> premiere2 = [{'élève' : 'Alan', 'note' : 19},{'élève' : 'Guido', 'note
    ' : 12}]
>>> premiere_reunion = premiere1 + premiere2
```

```
>>> premiere_reunion
[{'élève': 'Donald', 'note': 18}, {'élève': 'Ada', 'note': 20}, {'élève':
     'Alan', 'note': 19}, {'élève': 'Guido', 'note': 12}]
>>> premiere3 = [{'prénom' : 'Brian', 'nom' : 'Kernighan', 'note' : 14},{'
     prénom' : 'Linus', 'nom' : 'Torvalds', 'note' : 8}]
>>> premiere_reunion2 = premiere_reunion + premiere3
>>> prenom = [enreg['élève'] for enreg in premiere_reunion2]
Traceback (most recent call last):
......
KeyError: 'élève'
```

Exercice 1

1. Télécharger sur la page https://www.data.gouv.fr/fr/datasets/prenoms-a-rennes/ le fichier `prenoms_gf_rennes_zip`. Extraire l'archive, elle contient huit fichiers CSV dont le nom suit le format `Prénoms2008GF_Rennes.csv`, recensant les prénoms donnés aux enfants nés dans la ville de Rennes entre les années 2008 et 2015.

2. Ouvrir un fichier au hasard et lister les attributs de ces tables de données.

3. Ouvrir dans un IDE Python le fichier `TP_Fusion_Eleve.py` qui contient des fonctions `lecture_csv` et `ecriture_csv` de manipulation de fichiers CSV avec le module `csv` et des codes à compléter.

   Charger les tables contenues dans `Prénoms2011GF_Rennes.csv` et `Prénoms2010GF_Rennes.csv` avec l'instruction :

   ```
   prenom_11 = lecture_csv('Prénoms2011GF_Rennes.csv', ';', 'ISO8859')
   prenom_10 = lecture_csv('Prénoms2010GF_Rennes.csv', ';', 'ISO8859')
   ```

   On peut noter deux paramètres importants dans l'importation d'un fichier CSV : le délimiteur facile à déterminer en éditant le fichier (ici le symbole `';'`) et l'encodage (ici `'ISO8859'`) qui peut être obtenu avec la commande `file` sous Linux :

   ```
   fjunier@fjunier:~/TP-Fusion/Ressources$ file Prénoms2008GF_Rennes.
        csv
   Prénoms2008GF_Rennes.csv: ISO-8859 text, with CRLF line terminators
   ```

4. Tester un mauvais paramètre d'encodage en remplaçant `'ISO8859'` par `'utf8'` dans `lecture_csv`.

5. Compléter le code ci-dessous en respectant les spécifications données en `docstring` ou commentaires. On donne des postconditions pour vérifier les fonctions `reunion_table` et `prenom_max`.

   ```
   def reunion_table(table1, table2):
       """Paramètres : deux tables table1 et table2
       Valeur renvoyée : table concaténée de table1 et table2"""
   ```

```
    return ......................

def prenom_max(table):
    """Paramètres : table (ou réunion de tables) de prénoms
    Valeur renvoyée : tuple constitué de l'effectif maximum
        et du tableau des prénoms les plus fréquents"""
    effectif_max = int(table[0]['NBR'])
    histo = {table[0]['PRN'] : effectif_max}
    tab_max = [table[0]['PRN']]
    for enreg in table[1:]:
        prenom = enreg['PRN']
        effectif = int(enreg['NBR'])
        if prenom not in histo:
            histo[prenom] = ................
        else:
            histo[prenom] = ................
        if histo[prenom] > effectif_max:
            ......................
            ......................
        elif histo[prenom] == effectif_max:
            ......................
            ......................
    return (effectif_max, tab_max)

prenom_10_11 = reunion_table(prenom_10, prenom_11)
assert len(prenom_10) + len(prenom_11) == len(prenom_10_11)
#filtrer uniquement les prénoms féminins dans prenom_10_11
prenom_f_10_11 = [enreg for enreg in prenom_10_11
                  if ..............................................]
assert prenom_max(prenom_f_10_11) == (103, ['Louise'])
```

6. Exécuter le code ci-dessous. Quel point de vigilance lors de la réunion de tables est ainsi mis en évidence ?

```
#chargement de deux nouvelles tables
prenom_08 = lecture_csv('Prénoms2008GF_Rennes.csv', ';', 'ISO8859')
prenom_09 = lecture_csv('Prénoms2009GF_Rennes.csv', ';', 'ISO8859')
prenom_08_09 = reunion_table(prenom_08, prenom_09)

def clef_prenom(enreg):
    return enreg['PRN']

def clef_prenom_annee(enreg):
    return (enreg['PRN'], enreg['ANNAISS'])
```

```
assert sorted(prenom_09, key = clef_prenom_annee) == sorted(prenom_09,
    key = clef_prenom)
prenom_08_09_tri = sorted(prenom_08_09, key = clef_prenom_annee)
```

7. Charger les tables des prénoms en 2014 et compléter le code pour sélectionner uniquement les mois de naissance. Que peut-on remarquer ?

```
prenom_14 = lecture_csv('Prénoms2014GF_ Rennes.csv', ';', 'ISO8859')
# noter l'espace malencontreux dans le nom de fichier => erreur humaine
# sélectionner uniquement les mois de naissance en 2014
mois_14 = [..... for enreg in prenom_14]
assert mois_14[:4] == ['', '', '', '']
```

7. Exécuter le code ci-dessous. Quel point de vigilance lors de la réunion de tables est ainsi mis en évidence ?

   Peut-on quand même réunir les tables `prenom_14` et `prenom_15` dans `prenom_14_15`, et trier cette dernière par prénom puis par année ?

```
prenom_15 = lecture_csv('Prénoms2015GF_Rennes.csv', ';', 'ISO8859')
mois_15 = [enreg['MNAISS'] for enreg in prenom_15]
```

2 Jointure de tables

Point de cours 2

Si deux tables n'ont pas la même structure (différence sur certains attributs) mais partagent un ou plusieurs attributs (en nom et domaine de valeurs), on peut les fusionner en construisant leur **jointure**. On rapproche les enregistrements des deux tables qui ont la même valeur sur le(s) attibut(s) commun(s).

On donne ci-dessous un exemple de jointure de deux tables représentant les membres d'une plateforme d'entraînement à la programmation et les résultats d'un concours. La jointure s'obtient avec deux boucles imbriquées : on parcourt tous les couples constitués d'enregistrements de la première puis de la seconde table et on filtre à l'aide d'une conditionnelle. On peut écrire la jointure en une ligne avec un tableau défini en compréhension.

```
>>> membre = [{'nom' : 'knuth', 'prénom' : 'donald', 'pseudo' : 'knuth69'
    }, {'nom' : 'van rossum', 'prénom' : 'guido', 'pseudo' : 'guido42'}]
>>> concours = [{'pseudo' : 'knuth69', 'points' : 120}, {'pseudo' : '
    guido42', 'points' : 10}]
>>> def fusion_enreg(c, m):
...     return {'nom' : m['nom'], 'prénom' : m['prénom'], 'pseudo' : m['
    pseudo'], 'points' : c['points']}
>>> def fusion_tables(concours, membre):
...     jointure = []
```

```
...     for m in membre:
...         for c in concours:
...             if c['pseudo'] == m['pseudo']:
...                 jointure.append(fusion_enreg(c, m))
...     return jointure
>>> fusion_tables(concours, membre)
[{'nom': 'knuth', 'prénom': 'donald', 'pseudo': 'knuth69', 'points': 120},
      {'nom': 'van rossum', 'prénom': 'guido', 'pseudo': 'guido42', '
     points': 10}]
>>> def fusion_tables2(concours, membre):
...     return [fusion_enreg(c, m)
                for m in membre
                for c in concours
                if c['pseudo'] == m['pseudo']]
```

Exercice 2

On travaille toujours avec le fichier `TP_Fusion_Eleve.py` ouvert dans un IDE Python. On fournit deux tables dans les fichiers `'films.csv'` et `'realisateurs.csv'` dont on donne les premières lignes ci-dessous.

`films.csv` recense les films détenus en DVD par un cinéphile.

```
titre,réalisateur,année
La  vie est belle,Capra,1946
La  vie est belle,Benigni,1997
Beau-père,Blier,1981
```

`realisateurs.csv` recense des informations supplémentaires sur les réalisateurs des films de `'films.csv'`. Les tables sont séparées pour éviter les redondances car un même réalisateur peut avoir tourné plusieurs films.

```
nom,prénom,pays
Coppola,Sofia,USA
Coppola,Francis Ford,USA
Coppola,Roman,USA
Wilder,Billy,USA
```

1. Charger les deux tables dans les variables `films` et `realisateurs`.

```
films = lecture_csv('films.csv', ',', 'utf8')
realisateurs = lecture_csv('realisateurs.csv', ',', 'utf8')
```

2. On veut fusionner les tables films et réalisateurs par jointure sur l'attribut `'nom'` du réalisateur. Compléter le code ci-dessous en appliquant l'algorithme décrit dans le point de cours 2. On donne une postcondition pour vérifier la correspondance des deux fonctions de fusion.

```
def fusion_enregistrements(f, r):
    return {'titre' : f['titre'], 'année' : f['année'],
    'nom réalisateur' : r['nom'],'prénom réalisateur' : r['prénom'],
    'pays réalisateur' : r['pays']}

def fusion_tables(films, realisateurs):
    fusion = []
    for f in films:
        for r in realisateurs:
            ...................
            ...................
    return fusion

def fusion_tables_comprehension(films, realisateurs):
    return .......................................

jointure_f_r = fusion_tables(films, realisateurs)
jointure_f_r2 = fusion_tables_comprehension(films, realisateurs)
```

3. Exécuter les instructions ci-dessous. Quelle incohérence peut-on observer ? Explication ?

```
print([enreg for enreg in jointure_f_r if enreg['titre'] == 'CQ'])
print(len(jointure_f_r), len(films))
```

4. Exécuter les instructions ci-dessous. Quelle incohérence peut-on observer ? Explication ?

```
print([enreg for enreg in films if enreg['titre'] == 'Autant en
     emporte le vent'])
print([enreg for enreg in jointure_f_r if enreg['titre'] == 'Autant
     en emporte le vent'])
```

5. Éditer avec un éditeur de texte, les fichiers `films_idr.csv` et `realisateurs_id.csv`. On a rajouté dans le fichier des réalisateurs un identifiant unique (le numéro de l'enregistrement) qu'on a reporté dans le fichier des films à la place du nom du réalisateur.

   Compléter le code ci-dessous pour réaliser la fusion par jointure des tables chargées dans `films_idr` et `realisateurs_id`.

   Les incohérences repérées dans les questions précédentes sont-elles toujours présentes ? Quels sont les avantages de l'identifiant unique ?

```
films_idr = lecture_csv('films_idr.csv', ',', 'utf8')
realisateurs_id = lecture_csv('realisateurs_id.csv', ',', 'utf8')
def fusion_tables_id(films, realisateurs):
    return ...............................
jointure_f_r3 = fusion_tables_id(films_idr, realisateurs_id)
```

Point de cours 3

Reprenons l'exemple du point de cours 2 mais avec une personne supplémentaire dans la table `membre`, qui n'a pas participé au concours mais possède par hasard le même pseudo que `guido van rossum`. Si on fusionne les tables `membre` et `concours` par jointure sur l'attribut `'pseudo'` on obtient alors un doublon avec un enregistrement qui ne correspond à rien :

```
>>> concours = [{'pseudo' : 'knuth69', 'points' : 120}, {'pseudo' : '
    guido42', 'points' : 10}]
>>> membre = [{'nom' : 'knuth', 'prénom' : 'donald', 'pseudo' : 'knuth69'
    }, {'nom' : 'van rossum', 'prénom' : 'guido', 'pseudo' : 'guido42'},
    {'nom' : 'python', 'prénom' : 'monty', 'pseudo' : 'guido42'}]
>>> fusion_tables(concours, membre)[{'nom': 'knuth', 'prénom': 'donald', '
    pseudo': 'knuth69', 'points': 120}, {'nom': 'van rossum', 'prénom': '
    guido', 'pseudo': 'guido42', 'points': 10}, {'nom': 'python', 'prénom
    ': 'monty', 'pseudo': 'guido42', 'points': 10}]
```

Une solution est d'associer un **identifiant unique** à chaque enregistrement de la table `membre` (ou de ne pas autoriser de doublons dans le domaine de valeurs de l'attribut `'pseudo'`) et de l'insérer dans la table `concours`.

```
>>> concours = [{'ids': 1, 'pseudo' : 'knuth69', 'points' : 120}, {'id' :
    2, 'pseudo' : 'guido42', 'points' : 10}]
>>> membre = [{'id' : 1, 'nom' : 'knuth', 'prénom' : 'donald', 'pseudo' :
    'knuth69'}, {'id' : 2, 'nom' : 'van rossum', 'prénom' : 'guido', '
    pseudo' : 'guido42'}, {'id' : 3, 'nom' : 'python', 'prénom' : 'monty'
    , 'pseudo' : 'guido42'}]
>>> def fusion_tables3(concours, membre):
...     return [fusion_enreg(c, m) for m in membre for c in concours
                if c['id'] == m['id']]
>>> fusion_tables3(concours, membre)[{'nom': 'knuth', 'prénom': 'donald',
    'pseudo': 'knuth69', 'points': 120}, {'nom': 'van rossum', 'prénom':
    'guido', 'pseudo': 'guido42', 'points': 10}]
```

3 Synthèse

Synthèse

La **fusion** de tables peut se faire de plusieurs façons. Si les **tables** ont les mêmes **attributs** alors on peut les concaténer. Sinon, si les **tables** ont un ou plusieurs **attributs** communs, on peut réaliser une **jointure** qui rapproche les enregistrements qui ont les mêmes valeurs sur les attributs partagés. Avec les **bases de données**, étudiées en terminale, des opérations de jointure permettront de créer de nouvelles tables pour rassembler des informations issues de plusieurs tables.

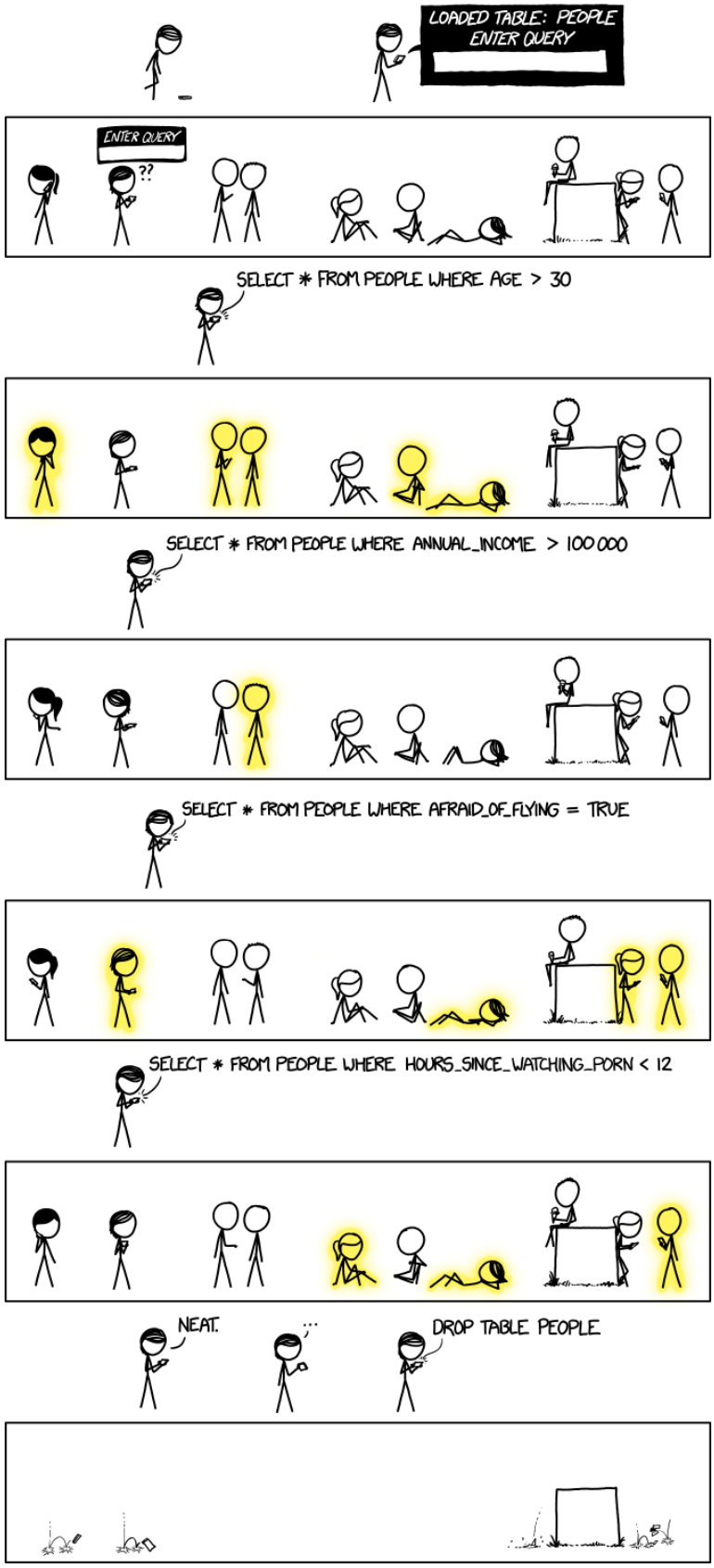

Figure 1: XKCD 1409 => SQL en route pour la terminale !