Lorsqu'on manipule des bases de données, on rassemble très rarement les données dans une seule table. Il est donc indispensable de savoir en mettre plusieurs en commun tout en respectant la logique des informations stockées.

Nous allons découvrir les notions de **réunion** et de **jointure** de tables.

1 Réunion de tables

On parle de **réunion de deux tables** lorsqu'on met bout à bout des tables possédant les mêmes colonnes.

En Python, cette opération correspond à la concaténation de ces dernières, réalisable avec l'opération `+`.

Par exemple, si on dispose de deux tables `elevesG1` et `elevesG2` contenant des dictionnaires de même clés, on peut les réunir dans une troisième table :

```
eleves = elevesG1 + elevesG2
```

`elevesG1`

nom	jour	mois	annee
Toto	12	11	2004
Titi	30	3	2004
Tutu	7	7	2004

`elevesG2`

nom	jour	mois	annee
Momo	5	2	2003
Mimi	25	7	2004
Mumu	9	9	2005

`eleves`

nom	jour	mois	annee
Toto	12	11	2004
Titi	30	3	2004
Tutu	7	7	2004
Momo	5	2	2003
Mimi	25	7	2004
Mumu	9	9	2005

À faire :

1. Écrire les fichiers CSV des tables `elevesG1` et `elevesG2` ;
2. importer ces tables dans un programme Python ;
3. définir dans ce programme la table `eleves`.

Attention : on veut parfois faire une réunion de deux tables dont l'une a plus de colonnes de l'autre. On pensera dans ce cas à retirer ces colonnes au préalable à l'aide d'une selection.

Dans l'exemple précédent, si la table `elevesG2` possède en plus une colonne `"ville"`, On pourra effectuer la réunion comme suit :

```
elevesG2_sans_ville = [
    {"nom":e["nom"], "jour":e["jour"], "mois":e["mois"], "annee":e["annee"]}
    for e in elevesG2
    ]

eleves = elevesG1 + elevesG2_sans_ville
```

2 Jointure de tables

Ajoutons à notre exemple une nouvelle table `notes` dans laquelle on garde en mémoire la moyenne des élèves et leurs appréciations :

`notes`

nom	note	appreciation
Toto	12	ok
Momo	15	très bien
Mumu	14	bien

À faire : Définir directement en Python (ou écrire le CSV et l'importer) la table `notes`.

On aimerait pouvoir générer une table contenant les informations à la fois des notes et des dates de naissance, étant entendu que les prénoms de la table `notes` correspondent aux mêmes prénoms que dans la table `eleves`.

Pour faire cela, on commence par écrire une fonction `fusion(e, n)` qui prend en argument deux lignes appartenant respectivement aux tables `eleves` et `notes` qui présentent le même attribut `"nom"` et qui renvoie une ligne contenant l'ensemble des informations des deux lignes :

```
def fusion(e, n):
    assert e["nom"] == n["nom"], "lignes incompatibles"
    return {
        "nom" : e["nom"],
        "jour" : e["jour"],
        "mois" : e["mois"],
        "annee" : e["annee"],
        "note" : n["note"],
        "appreciation" : n["appreciation"]
        }
```

À faire : Recopier le code de cette fonction.

La fonction `fusion` étant définie, on veut maintenant l'appeler sur **tous les couples `(e, n)` d'élèves et notes partageant le même nom.** On peut faire cela à l'aide d'une **double compréhention** :

```
eleves_notes = [
    fusion(e,n)
    for e in eleves
    for n in notes
    if e["nom"] == n["nom"]
    ]
```

À faire : Effectuer cette jointure.

3 Utilisation d'un identifiant

Dans notre exemple, on réussit à réunir les tables `notes` et `eleves` à l'aide d'un attribut commun `nom` (qui est ici un prénom). C'est évidemment une très mauvaise idée dans la mesure où plusieurs élèves peuvent partager le même nom.

il est bien sûr possible d'enrichir nos tables en rajoutant plus d'informations, comme par exemple nom de famille des élèves afin d'éviter ce problème, mais là encore il reste possible que deux élèves possède le même nom et prénom.

Plutôt que de faire cela, il est très courant en base de données de d'attribuer à chaque entité, un **identifiant unique** permettant de l'identifier. On s'arrange toujours en cas de réunions de tables ou d'ajout de lignes de conserver l'unicité de cet identifiant.

Cela permet à la fois d'être certain de pouvoir effectuer correctement des jointures, mais aussi de limiter les données stockée (ici un seul nombre permet d'identifier une personne).

Modifions nos tables pour mettre en oeuvre ce paradigme :

`eleves`

id	nom	jour	mois	annee
1	Toto	12	11	2004
2	Titi	30	3	2004
3	Tutu	7	7	2004
4	Momo	5	2	2003
5	Mimi	25	7	2004
6	Mumu	9	9	2005

`notes`

id	note	appreciation
1	12	ok
4	15	très bien
6	14	bien

À faire :

1. À l'aide d'un compteur et d'une boucle, ajouter aux dictionnaires de la table `eleves` des identifiants uniques.
2. Redéfinir de manière cohérente la table `notes`.
3. Réécrire de manière cohérente la fonction `fusion`.
4. Effectuer de nouveau la jointure des deux tables.