T3 - Recherche dans une table

7 juin 2021

Motivation, enjeux

Les données stockées dans une table peuvent être, et sont généralement, très nombreuses. Aussi, le simple fait de consulter ces données nécessite des outils de recherche.

Les fonctions proposés dans ce chapitre seront basé sur l'exemple d'une table `eleve` décrivant des élèves faisant un projet informatique. On y trouve leur prénom, leur date de naissance (jour, mois, année) et le nom de leur projet.

prenom	**jour**	**mois**	**annee**	**projet**
Toto	23	11	2002	Simulation météo
...	...	...	...	...

Présentation

Recherche simple et agrégations

Sélection

Généralisation, vers les bases de données

Recherche simple et agrégations : recherche par attribut clé

Dans le cadre d'une recherche en table, il n'est pas vraiment souhaitable de rechercher une ligne entière. A la place, on commence par rechercher une valeur d'un attribut donné qu'on précise.

Par exemple, on peut écrire une fonction de recherche permettant de savoir si un élève est présent dans la table :

```
1 def appartient(p, eleve):
2     """teste si un eleve est present dans la table"""
3     for e in eleve:
4         if e["prenom"] == p:
5             return True
6     return False
```

Recherche simple et agrégations : Recherche par attribut clé

On peut également renvoyer une information particulière, par exemple le projet de l'élève trouvé.

```
def projet_de(p, eleve):
    """renvoie le projet de l eleve"""
    for e in eleve:
        if e["prenom"] == p:
            return e["projet"]
```

Ou encore renvoyer toute la ligne :

```
def ligne(p, eleve):
    """renvoie les donnees de l eleve"""
    for e in eleve:
        if e["prenom"] == p:
            return e
```

Remarque : Ces deux fonctions renvoient `None` si le prénom n'est pas dans la table.

Recherche simple et agrégations : fonctions d'agrégation

On parle de fonction d'agrégation pour désignés des calculs effectués sur un ou plusieurs attributs de tous les entités d'une table. Les plus courantes sont les occurrences, la somme, la moyenne, le minimum et le maximum.

Exemple de fonction d'occurrence :

```
1 def eleves_nes_en(eleve, a):
2     """renvoie le nombre d eleve nes en l annee a"""
3     nb = 0
4     for e in eleve:
5         if e["annee"] == a:
6             nb+=1
7     return nb
```

Exemple de fonction de moyenne :

```
1 def annee_moyenne(eleve):
2     """renvoie l annee de naissance moyenne des eleves"""
3     somme = 0
4     for e in eleve:
5         somme += e["annee"]
6     return somme/len(eleve)
```

Présentation

Recherche simple et agrégations

Sélection

Généralisation, vers les bases de données

Sélection : sélection de lignes

Jusqu'à maintenant on a considéré que nos fonctions de recherche ne renvoyaient qu'un seul élément. Or généralement, plusieurs entités peuvent partager une même valeur sur un attribut donné (même prénom, même date de naissance, même projet, etc.).

Ainsi, plutôt que de renvoyer les données d'une seule entité lors d'une recherche, on peut renvoyer la sous-table des entités présentant la valeur de l'attribut recherché ou plus généralement dont les attributs passent un certain test, on parle de **sélection par ligne**.

```
1 #les eleves nes en 2002
2 t1 = [e for e in eleve if e["annee"]==2002]
3
4 #les eleves nes avant 2002
5 t2 = [e for e in eleve if e["annee"]<=2002]
6
7 #les eleves dont le prenom commence par A
8 [e for e in eleve if e["prenom"][0]=='A']
```

Sélection : sélection de lignes et de colonnes

Une sélection peut également s'effectuer en ne gardant que certains attributs.

```
#noms de eleves nes en 2002
[{"nom":e["nom"]} for e in eleve if e["annee"]==2002]

#noms et projets des eleves nes avant 2002
t2 = [{"nom":e["nom"], "projet":e["projet"]} for e in eleve
    if e["annee"]<=2002]

#annees de naissance des eleves dont le nom commence par A
[{"annee":e["annee"]}}for e in eleve if e["prenom"][0]=='A']
```

Présentation

Recherche simple et agrégations

Sélection

Généralisation, vers les bases de données

Généralisation, vers les bases de données : fonctions générales

Les fonctions et commandes vues dans ce chapitre étaient généralement définies dans le cadre de notre exemple. Il est cependant possible d'en écrire des plus générales, permettant de manipuler des tables quelconques.

```
1 def select(table, att_list, test):
2     return [{att:e[att] for att in att_list} for e in table
      if test(e)]
3
4 def table_sum(table, att):
5     return sum([e[att] for e in table])
6
7 def count(table):
8     return len(table)
```

Généralisation, vers les bases de données : le langage SQL

SQL est un langage permettant d'administrer et d'interroger des bases de données, c'est à dire un ensemble de tables liées entre elles par des relations logiques.

Il permet notamment de formuler des **requêtes** permettant de recueillir des informations similaires à celles vues dans ce chapitre. Par exemple, pour extraire les mêmes informations que les trois commandes précédentes, on peut écrire en SQL :

```sql
SELECT nom FROM eleve WHERE annee = 2002 ;
SELECT nom, projet FROM eleve WHERE annee <= 2002 ;
SELECT annee FROM eleve WHERE prenom LIKE "A%" ;
```

L'étude des bases de données et du langage SQL sont au programme de terminale NSI.