Introduction à la notion de bases de données

Ces dernières années, le concept de bases de données s'est développé en raison notamment du nombre croissant d'informations à traiter en temps réel. On présentera simplement ce qu'on entend par bases de données et nous verrons comment opérer sur ces bases : des requêtes simples dans le langage SQL.

1 Représentation d'une base de données et premières opérations

1.1 Table de données et schéma relationnel

Si on souhaite regrouper une série de données dans le langage Python, on peut toujours utiliser les structures classiques : listes, tuples, ensembles, dictionnaires, tableaux... Malheureusement, toutes ces structures de données ne nous permettront pas de gérer des données plus complexes, présentant un grand nombre d'entrées.

Par exemple, si on souhaite gérer l'ensemble des notes de colles de toute la classe, on préfèrera utiliser un **modèle relationnel**, c'est à dire un ensemble de données aux entrées multiples aussi appelé **table de données** :

table des notes

id	**note**	élève	semaine	nom du prof	matière
s7Le1m	**15**	Sarah	7	Didier	maths
s7Le2m	**17**	Julie	7	Didier	maths
s7Le3m	**15**	Racha	7	Didier	maths

Les colonnes de la table sont appelées **attributs** et pour chaque colonne, on aura l'ensemble des valeurs prises par ces attributs. Les lignes de la table désignent les **relations** et pour chaque ligne, on aura un ensemble de valeurs associées à une note.

Généralement, on cherchera à repérer rapidement un **identifiant**, appelé aussi **clé primaire** qui est unique dans la table et qui nous permettra d'extraire les données d'une seule ligne à la fois. Cela permet en effet d'éviter les erreurs lors de l'extraction des notes, d'un calcul, les fautes de frappe... même si celui-ci n'est pas obligatoire.

De la même façon, nous pourrions construire la table des élèves, ou encore celle des colleurs :

table des élèves

id	**nom**	prénom	téléphone	âge
1	**Pereira**	Zoé	0675593451	20

table des colleurs

id	**nom**	matière	téléphone	adresse email
10	**Didier**	maths	0650143207	bgdu10@gmail.com

L'ensemble de toutes ces tables constitue alors une **base de données** associées à la classe et elle sera parfois représenter de la façon suivante :

classe de MP

table des notes		*tables des élèves*		*table des colleurs*	
id	TEXT	id	INTEGER	id	INTEGER
note	INTEGER	nom	TEXT	nom	TEXT
élève	TEXT	prénom	TEXT	téléphone	TEXT
semaine	INTEGER	téléphone	TEXT	mail	TEXT
nom du prof	TEXT	âge	INTEGER		
matière	TEXT				

La gestion d'une base de données relationnelles consiste simplement à gérer ces différentes tables de données pour en organiser les contenus.
D'ailleurs, ces derniers seront souvent appelés par les **attributs** donnés à chaque colonne d'une table ou en cas de conflits, on pensera à **préfixer les noms des attributs par la table** dont ils proviennent : par exemple, on pourra appeler les attributs *eleves*.nom ou *colleurs*.nom.

1.2 Opérateurs relationnels

Afin d'illustrer les différentes opérations sur des tables de données, on considère deux tables possédant le même schéma relationnel, c'est à dire le même nombre d'attributs, les mêmes noms d'attributs et mêmes domaines. Par exemple :

table des profs de MP

nom	matière	ancienneté
Didier	maths	14
Pertreux	physique	4
Andry	anglais	8
Desloges	SI	8

table des profs de PSI

nom	matière	ancienneté
Golse	physique	20
Rharif	maths	25
Andry	anglais	8
Desloges	SI	8

Notons R_1 la table des profs de MP et R_2 la table des profs de PSI. On peut alors appliquer les opérations ensemblistes usuels :

- la **réunion** $R_1 \cup R_2$ définie par la table :

nom	matière	ancienneté
Didier	maths	14
Pertreux	physique	4
Andry	anglais	8
Desloges	SI	8
Golse	physique	20
Rharif	maths	25

- l'**intersection** $R_1 \cap R_2$ définie par la table :

nom	matière	ancienneté
Andry	anglais	8
Desloges	SI	8

- la **différence** $R_1 - R_2$ qui renvoie l'ensemble des éléments de R_1 qui n'appartiennent pas à R_2 :

nom	matière	ancienneté
Didier	maths	14
Pertreux	physique	4

Par ailleurs, il est aussi tout à fait possible d'extraire des données d'une table : on parle de **projection sur des attributs**. Pour cela, on va sélectionner les données par colonne.

Concrètement, si on souhaite extraire le nom et la matière des professeurs de première année, on applique la projection de la table $R_1 \cup R_2$ sur les attributs $nom, matiere$ de sorte que :

$\Pi_{nom,matiere}(R_1 \cup R_2) =$

nom	matière
Didier	maths
Perteux	physique
Andry	anglais
Desloges	SI
Golse	physique
Rharif	maths

1.3 Cas particulier de la sélection

Bien entendu, on peut affiner cette sélection et extraire aussi des données suivant les valeurs des attributs : on parle alors de **sélection suivant une ou plusieurs conditions**.

Concrètement, si on souhaite extraire le nom des profs de maths ou de physique, il suffit de sélectionner dans la table $R_1 \cup R_2$ les lignes satisfaisant la condition $matiere =' maths'$ d'une part ou les lignes satisfaisant la condition $matiere =' physique'$ d'autre part :

$\sigma_{matiere='maths'}(R_1 \cup R_2) \cup \sigma_{matiere='physique'}(R_1 \cup R_2) =$

nom	matière	ancienneté
Didier	maths	14
Rharif	maths	25
Pertreux	physique	4
Golse	physique	20

Remarques

1. Bien entendu, on pouvait aussi réunir les lignes de R_1 et R_2 satisfaisant la seule condition :

   $$matiere \in \{'maths','physique'\} \text{ ou encore } matiere =' maths' \text{ ou } matiere =' physique'.$$

   La seule exigence, c'est de définir une condition booléenne sur la valeur des attributs pour effecteur cette sélection.

2. Si par la suite, on veut extraire l'ancienneté des collègues concernés, alors on composera par une projection... en fait, on combinera très souvent les deux opérations sélection/projection afin d'extraire les données souhaitées.

Exemple 1 On considère deux bulletins météo dont on peut lire les informations suivantes :

bulletin n°1

domaine	station	altitude	enneigement
Espace Killy	Tignes	1550	250
Espace Killy	Val d'Isère	1850	280
Trois Vallées	Val Thorens	2300	350
Trois Vallées	Courchevel	1100	150
Trois Vallées	Les Ménuires	1850	180

bulletin n°2

domaine	station	altitude	enneigement
Paradiski	Champagny	2000	210
Paradiski	La Plagne	1250	180
Paradiski	Les Arcs	1600	200

On note B_1 et B_2 les tables associées. A l'aide des opérateurs relationnels, expliciter les opérations permettant d'obtenir les informations suivantes :

1. la liste des domaines mentionnés dans ces deux bulletins ;
2. toutes les informations concernant le domaine des Trois Vallées ;
3. le nom des stations dont l'enneigement est supérieur ou égal à 200 cm ;
4. le nom et l'altitude des stations situées au dessus de 2000 m.

2 Gestion d'une base de données relationnelles

La plupart du temps, lorsqu'on travaille sur une base de données relationnelles, on n'y a pas accès directement. En effet, pour éviter que les données initiales soient à la portée de tous ou tout simplement écrasées, elles sont stockées sur un serveur indépendant. Les opérations exécutées par l'utilisateur se font alors à travers une interface logicielle.

Toutes ces opérations sont en fait des **requêtes** et on apprendra à utiliser un langage standard aux bases de données, le **langage SQL** pour *Structured Query Language*.

2.1 Une syntaxe très simple

Une fois connectée à la base, on peut alors appliquer tous les opérateurs relationnels qui ont été présentés précédemment. On pourra retenir en particulier que toutes ces requêtes possèdent une structure commune basée sur les commandes :

`SELECT` *attribut*(s) `FROM` *table* `WHERE` *condition*(s)

On considère encore la base de données des profs de MP et de PSI, et on traduit ici quelques exemples de requêtes dans le langage SQL :

Opérateur relationnel	Commande SQL
la réunion $R_1 \cup R_2$	`SELECT * FROM mp UNION SELECT * FROM psi`
l'intersection $R_1 \cap R_2$	`SELECT * FROM mp INTERSECT SELECT * FROM psi`
la différence $R_1 - R_2$	`SELECT * FROM mp WHERE nom NOT IN (SELECT nom FROM psi)`
la projection sur les attributs $nom, matiere$	`SELECT nom,matiere FROM (SELECT * FROM mp UNION SELECT * FROM psi)`
la sélection des profs de maths ou physique	`SELECT * FROM (SELECT * FROM mp UNION SELECT * FROM psi) WHERE matiere = 'maths' or matiere ='physique'`

Remarque En fonction des logiciels, on pourra entrer ces requêtes en ligne en respectant simplement les majuscules pour les commandes SQL. Par contre, d'autres logiciels exigent de sauter des lignes à chaque groupe de commandes... on s'adaptera et il est vrai qu'au concours, c'est mieux de présenter des requêtes lisibles !

Exemple 2 On reprend les données météo de l'exemple précédent. Les tables associées sont encore notées B_1 et B_2.

Dans le langage SQL, déterminer les requêtes permettant d'extraire les informations suivantes :

1. la liste des domaines mentionnés dans ces deux bulletins ;
2. toutes les informations concernant le domaine des Trois Vallées ;
3. le nom des stations dont l'enneigement est supérieur ou égal à 200 cm ;
4. le nom et l'altitude des stations situées au dessus de 2000 m.

2.2 Jointure et fonctions d'agrégation

Jusqu'à maintenant, nous avons travaillé sur des tables possédant les mêmes attributs, mais il est aussi tout à fait possible de croiser les informations provenant de tables différentes, on parle alors de **jointure**.

Par exemple, si on considère les tables suivantes :

table des profs de MP

nom	matière	ancienneté
Didier	maths	14
Pertreux	physique	4
Andry	anglais	8
Desloges	SI	8

table des coordonnées

mail	nom	téléphone
perrine.andry@ac-reims.fr	Andry	0658321045
etienneetienne@phub.com	Pertreux	0654895312
bgdu10@gmail.com	Didier	0650143207
roidutorseur@laposte.net	Desloges	0621400847

on peut construire le **produit cartésien** de ces tables, mais cela nous donnerait 4 x 4 possibilités d'association avec des lignes qui ne seraient pas compatibles.
On cherche donc à associer ces tables en éliminant les lignes inutiles et ceci afin que les informations recoupées correspondent à un seul et même individu : on parle ici de **jointure**.

Pour cela, il nous faudra préciser l'attribut permettant de faire la jointure des deux tables. Ainsi, si on réalise la jointure de nos deux tables encore notées R_1 et R_2 suivant le nom, il vient :

$R_1 \underset{R_1.nom=R_2.nom}{\bowtie} R_2 =$

nom	matière	ancienneté	**mail**	nom	téléphone
Didier	maths	14	**bgdu10@gmail.com**	Didier	0650143207
Pertreux	physique	4	**etienneetienne@phub.com**	Pertreux	0654895312
Andry	anglais	8	**perrine.andry@ac-reims.fr**	Andry	0658321045
Desloges	SI	8	**roidutorseur@laposte.net**	Desloges	0621400847

En langage SQL, on applique la même structure en précisant l'attribut grâce auquel on réalise la jointure :

```
SELECT *
FROM mp
JOIN coordonnees ON mp.nom=coordonnees.nom
```

Bien entendu, on peut aussi affiner la requête pour éviter les colonnes redondantes et par exemple :

```
SELECT mp.nom,mp.matiere,coordonnees.mail,coordonnees.telephone
FROM mp
JOIN coordonnees ON mp.nom=coordonnees.nom
```

Remarques

1. On peut aussi réaliser des jointures sur plusieurs tables : il suffit de répéter les commandes JOIN... ON... autant de fois que nécessaire:
   ```
   SELECT *
   FROM mp
   JOIN coordonnees ON mp.nom=coordonnees.nom
   JOIN table3 ON mp.nom=table3.clef3 JOIN table4 ON mp.nom=table4.clef4 ...
   ```

2. Quand le nom d'une colonne n'est pas expicite, on peut la renommer en imposant un alias à la sélection avec la commande AS :
   ```
   SELECT mp.nom AS nom du prof,coordonnees.telephone AS tel
   FROM mp
   JOIN coordonnees ON mp.nom=coordonnees.nom
   ```

3. Toutes ces commandes en langage SQL nous permettent d'extraire des informations contenues dans une base de données relationnelle, mais il existe aussi d'autres commandes très pratiques facilitant la manipulation de ces données : ce sont les **fonctions d'agrégation qui s'appliquent à des groupes de lignes**...

   On essaiera d'en connaître quelques unes, mais attention, leur utilisation est loin d'être aisée :

Commande SQL	Résultat
`SELECT nom,matiere FROM mp ORDER BY anciennete ASC/DESC`	renvoie les noms et matières des profs de MP triés par ancienneté par ordre croissant ou décroissant
`SELECT COUNT(*) FROM mp`	renvoie le nombre de lignes de la table des profs de MP
`SELECT COUNT(DISTINCT matiere) FROM mp GROUP BY matiere`	renvoie le nombre d'éléments (distincts ou non) dans chaque matière quand on regroupe les lignes par matière.
`SELECT MAX/MIN(anciennete) FROM mp`	renvoie l'ancienneté maximale/minimale
`SELECT AVG/SUM(anciennete) FROM mp`	renvoie la moyenne/somme de l'ancienneté

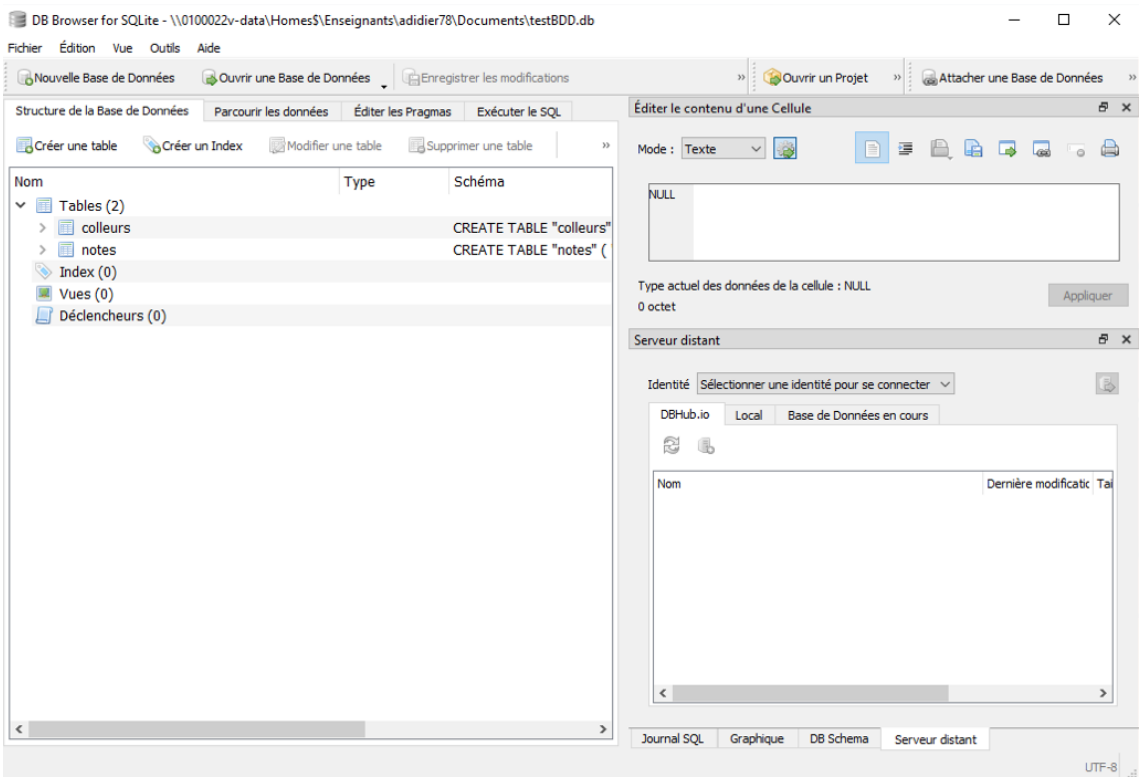
Pour finir, on retiendra que d'une façon générale, toutes nos requêtes suivent toujours la même syntaxe :

```
SELECT .....
FROM .....
WHERE .....
   GROUP BY .....
   HAVING .....
ORDER BY .....
```

Attention, si on souhaite filtrer les réponses après avoir utilisé des fonctions d'agrégation, il faudra prendre l'habitude d'utiliser les commandes `GROUP BY` et `HAVING` pour que celles-ci puissent être appliquées à l'agrégat de données obtenu.

3 Un premier exemple de requêtes dans le langage SQL

Pour mieux comprendre les différentes insctructions dans le langage SQL, on va essayer de traiter quelques requêtes à l'aide du logiciel **SQLite** :



Une fois la base de données ouvertes, on peut naturellement, à l'aide des onglets proposés, **parcourir les données** ou **exécuter des requêtes dans le langage SQL**. Par exemple, on considère la base de données suivante constituée de deux tables : celles des *élèves* et des *colleurs* :

élèves

id	groupe	nom	note	statut	prof
1	A	Nathoun Baroun	8		Antonio Didy
2	A	Kevin Weliachew	7	cinqdemi	Antonio Didy
3	A	Widgy Bigmuscles	20	cinqdemi	Antonio Didy
4	B	Major Oudini	15	cinqdemi	Vincent Molot
5	B	Zozo Gourmandos	14	cinqdemi	Vincent Molot
6	B	Marwa Amazing	14		Vincent Molot
7	C	Sacha Lébébé	16		Dominik Alair
8	C	Etienne Blanvillet	13		Dominik Alair
9	C	Océane Claudon	17		Dominik Alair
10	D	Martin Weliachew	15	cinqdemi	Antonio Didy
11	D	Elisa Seagull	15	cinqdemi	Antonio Didy
12	D	Wincent Mondoudou	11		Antonio Didy

colleurs

id	nom	âge	tel	groupe
1	Dominik Alair	62	24568000	C
2	Vincent Molot	38	0603168595	B
3	Antonio Didy	43	0650143208	A
4	Antonio Didy	43	0650143208	D

1. Ecrire une requête permettant d'afficher toutes les données de la table *élèves* :

2. Ecrire une requête permettant d'obtenir le nombre de groupes dans la table *élèves* :

3. Ecrire une requête permettant d'obtenir le nombre de colleurs en activité (avant 60 ans) :

4. Ecrire une requête permettant d'obtenir le nom et la note des élèves 5/2 :

5. Ecrire une requête permettant d'obtenir le nom et la note des élèves rangés par ordre décroissant :

6. Ecrire une requête permettant d'obtenir le nom des élèves parmi les 3/2 qui ont eu plus que la moyenne des notes :

7. Ecrire une requête permettant d'obtenir le nom et la note des élèves qui ont eu Monsieur Didy et qui ont eu plus de 10, rangés par ordre croissant des notes obtenus :

8. Ecrire une requête permettant d'obtenir la moyenne des notes des élèves interrogés pour chaque colleur :

9. Ecrire une requête permettant d'obtenir le nom des élèves qui ont éventuellement majoré, leurs notes mais aussi le nom du colleur associé et son âge :

10. Ecrire une requête permettant d'obtenir le nom et les coordonnées des colleurs qui ont interrogé les 5/2 :

11. Ecrire une requête permettant d'obtenir les groupes dont la moyenne par groupe est > 13 :

12. Ecrire une requête permettant d'obtenir le nombre d'élèves interrogés par colleur rangés dans l'ordre décroissant du nombre d'élèves interrogés :