Les bases de données TP Zoo[1]

Résumé :

Voilà une mise en œuvre d'une base de données autour d'un jardin zoologique. Cela nous permettra d'appliquer des requêtes élaborées telles que les jointures ou opérations ensemblistes.

L'implémentation en Python est également proposée.

Le plan du zoo[2]

Sommaire

[1] Ce travail s'inspire de l'activité base de données Zoo du DIU-EIL de l'Université de Grenoble.
[2] https://www.parczoologiquedeparis.fr/fr/plan-du-zoo-et-parcours-2363

1 Le zoo

Le directeur d'un Zoo a informatisé la gestion de son établissement.

Dans ce Zoo, on trouve des animaux répertoriés par type (lion, léopard, girafe, escargot, ...). Chaque animal possède un nom (Charly, Arthur, Enzo, ...) qui l'identifie de façon unique, un type (ou race), un type de cage (fonction) requis, une date de naissance et un pays d'origine. On retient également les maladies que chaque animal a contractées depuis son arrivée au Zoo, ainsi que le nombre de ses maladies.

Les animaux sont logés dans des cages. Chaque cage peut recevoir un ou plusieurs animaux. Certaines cages peuvent être inoccupées. Une cage correspond à une certaine fonctionnalité et ne permet de ne recevoir que des animaux compatibles. Une cage est identifiée par un numéro, elle est située dans une allée, identifiée aussi par un numéro. Des animaux de types différents ne peuvent pas cohabiter dans une même cage.

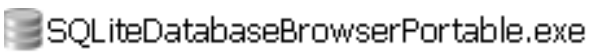
Les employées du Zoo entretiennent les cages et soignent les animaux. Chaque personne est identifiée par son nom, et on connaît la ville où elle réside. Elle est aussi spécialiste de tel ou tel type de cages.

Les personnes sont soit gardien, soit responsable. Les affectations, gardien ou responsable, doivent être compatible avec les spécialités de chacun. Un gardien s'occupe d'une ou de plusieurs cages, et un responsable a la charge de toutes les cages d'une ou de plusieurs allées. Une allée est supervisée par un seul employé et toute cage occupée par au moins un animal est gardée par au moins un gardien.

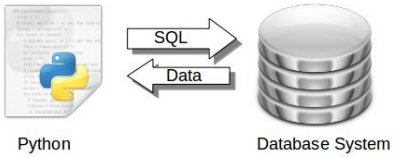
2 Création de la base

2.1 Le langage SQL

La base de données est créée avec des instructions MySQL. Les instructions sont contenues dans un fichier et celui-ci est exécuté soit via le logiciel SQLiteDatabaseBrowser soit via Python nous allons étudier les deux possibilités.[3]



Mais auparavant intéressons-nous à la syntaxe SQL utilisée.



[3] Illustration issue du site https://ichi.pro/fr/comment-importer-un-fichier-csv-dans-une-base-de-donnees-mysql-a-l-aide-de-python-133156956822579

Création d'une table

Voilà quelques possibilités de syntaxes à noter comment la clé primaire est déclarée dans chacun des cas :

```
CREATE TABLE LesCages(
     noCage int,
     fonction varchar(100),
     noAllee int,
     CONSTRAINT PK_Cages PRIMARY KEY(noCage)
);
```

```
CREATE TABLE Coordonnees(
    idCoord integer PRIMARY KEY,
    nomSalle  varchar2(40),
    Adresse   varchar2(40),
    Telephone varchar2(15)
);
```

Voilà maintenant pour les clés étrangères :

```
CREATE TABLE LesMaladies(
  nomA varchar(100),
  nomM varchar(100),
  CONSTRAINT PK_Maladies PRIMARY KEY(nomA, nomM),
  CONSTRAINT FK_Maladies_Animaux FOREIGN KEY (nomA) REFERENCES LesAnimaux(nomA)
);
```

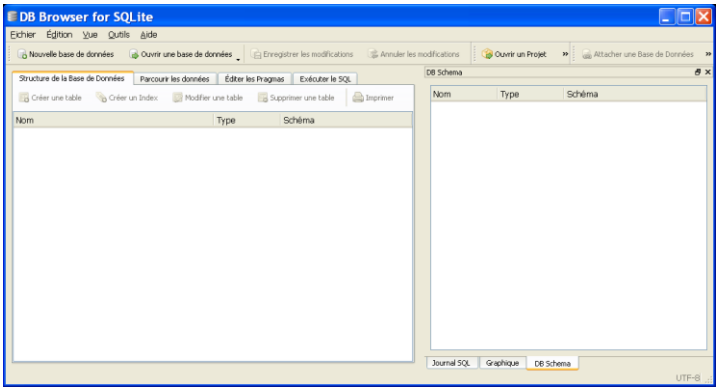
```
CREATE TABLE Acteur  (
    idActeur integer PRIMARY KEY,
    nomActeur varchar2(40),
    anneeNaissance integer,
    idFilm  integer,
    CONSTRAINT Acteur_C1 FOREIGN KEY (idFilm) REFERENCES Film (idFilm)
);
```

Et enfin l'insertion des données

```
INSERT INTO LesCages VALUES (11,'fauve', 10);
INSERT INTO LesCages VALUES (1,'fosse', 1);
```

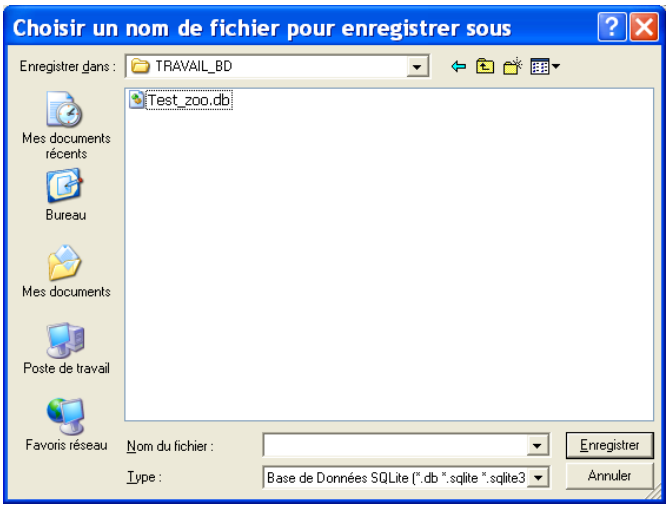
2.2 Création de la base de données avec SQLiteDatabaseBrowser

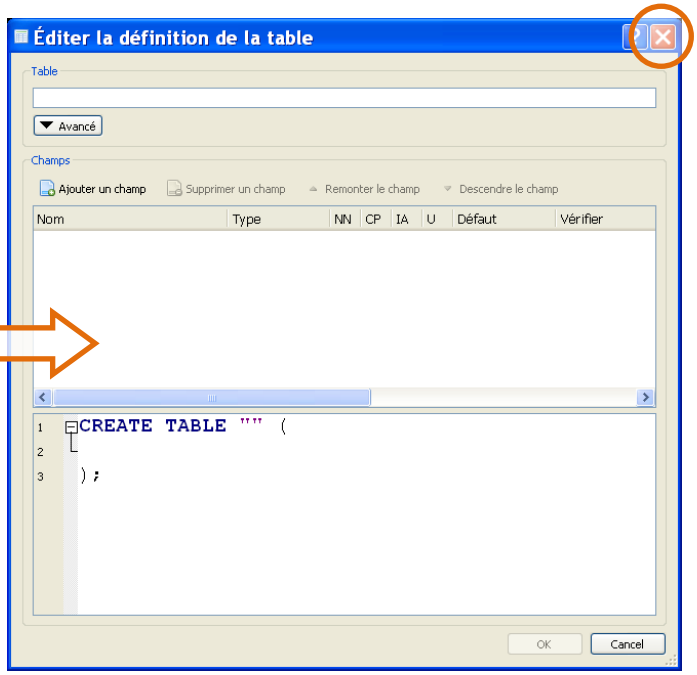
Ouvrir le logiciel



Choisir nouvelle base de données

Indiquez le dossier et le nom de la future base

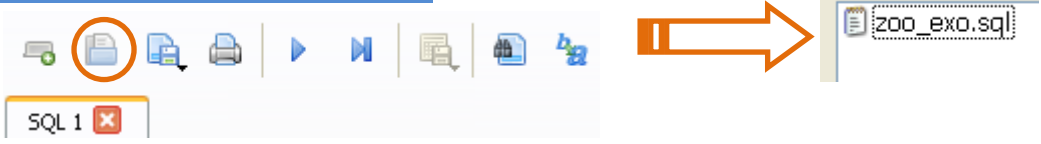




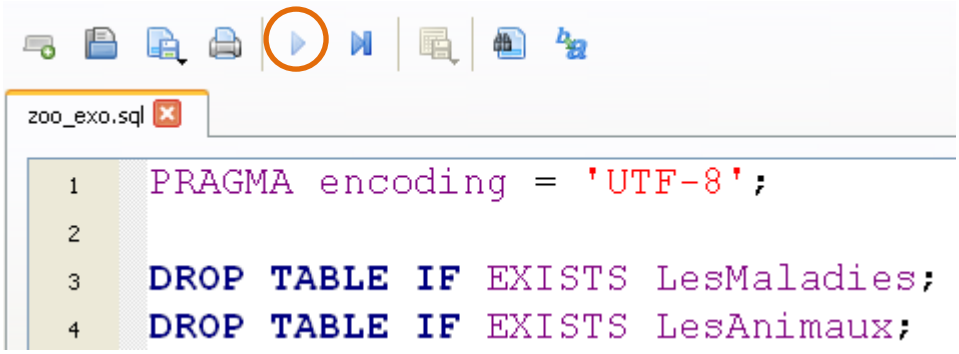
La fenêtre ci-contre s'ouvre on ne va pas l'utiliser il faut donc la fermer.

Choisir exécuter le SQL

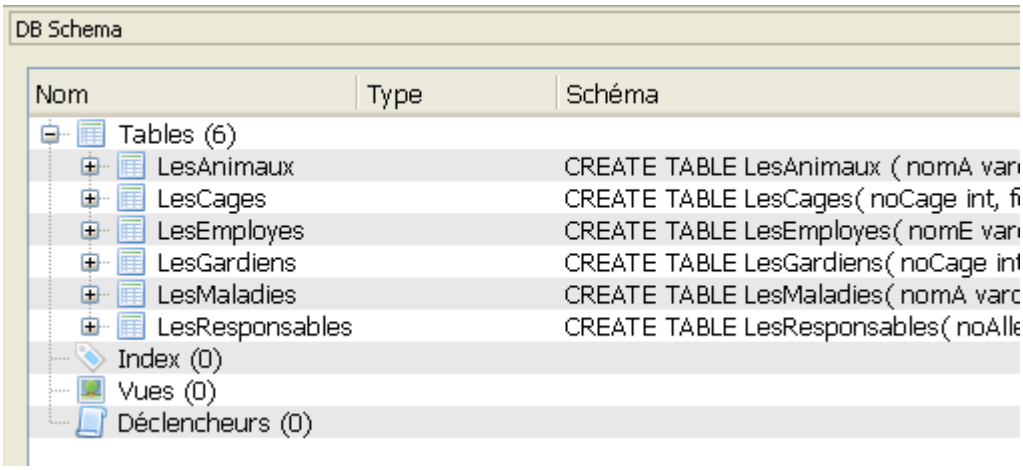
Puis importer un fichier SQL



Exécuter les instructions



La base de données est créée



Il ne reste plus qu'à la sauvegarder.

2.3 Création de la base avec Python

Python permet avec le module sqlite3 la création et l'accès aux bases de données. Les étapes à suivre sont les suivantes :

Importation de la bibliothèque

```
import sqlite3
```

Connexion à la base de données

```
conn = sqlite3.connect(db_file)
```

On accède ensuite à la base avec la création d'un objet curseur

```
cursor = conn.cursor()
```

Lecture du fichier texte contenant les instructions au format SQL

```
# Lecture du fichier et placement des requetes dans un tableau
createFile = open(file, 'r', encoding="UTF-8")
createSql = createFile.read()
createFile.close()
sqlQueries = createSql.split(";")
```

Attention : notez le décodage UTF-8.

Exécution des instructions

```
for query in sqlQueries:
    cursor.execute(query)
```

Forcer la mise à jour de la base

```
conn.commit()
```

Fermeture de la connexion

```
conn.close()
```

Vous pouvez tester la création de la base avec testzoo_creation.py

2.4 Gérer des requêtes avec Python

Nous avons déjà vu dans le document précédent l'utilisation de SQLiteDatabaseBrowser pour la gestion directe de requêtes sur une base. Voyons maintenant comment le réaliser avec Python.

Principe d'exécution d'une requête

```
def select_tous_animaux(conn):
    """
    :param conn: objet connexion
    :return:
    """
    cur = conn.cursor()
    cur.execute("SELECT * FROM LesAnimaux")

    rows = cur.fetchall()

    for row in rows:
        print(row)
```

```
2. Liste de tous les animaux
('Charly', 'male', 'lion', 'Kenya', 1999, 12)
('Arthur', 'male', 'ours', 'France', 2000, 1)
('Chloé', 'femelle', 'pie', 'France', 2001, 3)
('Milou', 'male', 'leopard', 'France', 2013, 11)
('Tintin', 'male', 'leopard', 'France', 2013, 11)
('Charlotte', 'femelle', 'lion', 'Kenya', 2012, 12)
('Huan', 'femelle', 'panda', 'Chine', 2005, 1)
('Lola', 'femelle', 'lion', 'Kenya', 1999, 5)
('Tola', 'femelle', 'ours', 'Espagne', 2003, 1)
('Taï Lung', 'femelle', 'leopard', 'Chine', 2006, 5)
('Yuang Meng', 'male', 'panda', 'France', 2015, 1)
```

Nous retrouvons la création du curseur sur la base, puis la méthode *fetchall* qui permet de récupérer d'un coup l'ensemble du résultat de la requête.

2.5 Ajouter et supprimer des enregistrements

Ajouter un nouvel enregistrement

```
INSERT INTO LesCages VALUES (13,'grande volière',2);
```

Table : LesCages

	noCage	fonction	noAllee
	Filtre	Filtre	Filtre
1	11	fauve	10
2	1	fosse	1
3	2	aquarium	1
4	3	petits oiseaux	2
5	4	grand aquarium	1
6	12	fauve	10
7	5	fauve	10
8	13	grande volière	2

Modifier un enregistrement

```
UPDATE LesCages SET noCage=15, noAllee=3 WHERE noCage=13;
```

Table : LesCages

	noCage	fonction	noAllee
	Filtre	Filtre	Filtre
1	11	fauve	10
2	1	fosse	1
3	2	aquarium	1
4	3	petits oiseaux	2
5	4	grand aquarium	1
6	12	fauve	10
7	5	fauve	10
8	15	grande volière	3

Table : LesCages

	noCage	fonction	noAllee
	Filtre	Filtre	Filtre
1	11	fauve	10
2	1	fosse	1
3	2	aquarium	1
4	3	petits oiseaux	2
5	4	grand aquarium	1
6	12	fauve	10
7	5	fauve	10

Supprimer un enregistrement

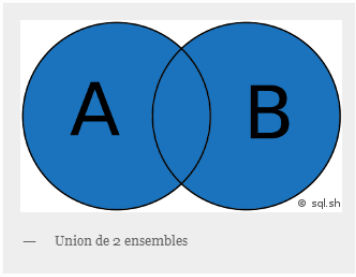
```
DELETE FROM LesCages WHERE noCage=15;
```

IMPORTANT : les ajouts ou modification des tables doivent respecter les contraintes exprimées lors de la création de la base de données pour respecter l'intégrité référentielle des données à tout moment. Cela peut conduire à un ordonnancement des opérations.

3 Approfondir les requêtes par l'exemple

3.1 Les opérations ensemblistes[4]

Union



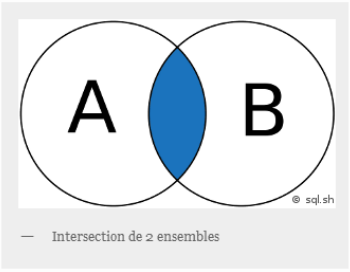
Obtenir tous les éléments qui correspondent à la fois à l'ensemble A et l'ensemble B.

	noCage
1	1
2	2
3	4
4	5
5	12

Donner les cages de l'allée 1 et celles (de n'importe quelle allée) qui contiennent un lion

```
SELECT noCage FROM LesCages WHERE noAllee=1
UNION
SELECT noCage FROM LesAnimaux WHERE typeA='lion';
```

Intersection



Obtenir tous les éléments qui correspondent simultanément à l'ensemble A et l'ensemble B.

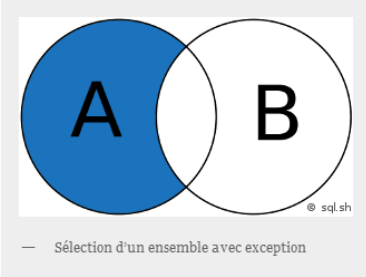
Donner les noms des animaux qui ont eu la grippe et une angine

```
SELECT nomA FROM LesMaladies WHERE nomM='grippe'
INTERSECT
SELECT nomA FROM LesMaladies WHERE nomM='angine';
```

	nomA
1	Huan

Différence

Selon les moteurs de SGBD on a deux soit *minus* soit *except* (sqlite3)



Obtenir tous les éléments qui appartiennent à l'ensemble A exclusivement.

[4] Les illustrations sont issues du site https://sql.sh/

Donner les numéros des cages qui ne sont pas gardées

```
SELECT noCage FROM LesCages
EXCEPT
SELECT noCage FROM LesGardiens;
```

	noCage
1	2
2	4

3.2 Les jointures

Les jointures permettent de travailler sur plusieurs tables.

Le produit cartésien

Le produit cartésien met en relation tous les éléments de la table A avec tous les éléments de la table B. Attention sur les grosses tables le résultat peu être très grand. Il sera utilisé en combinaison avec des processus de sélection pour ne conserver que certains résultats.

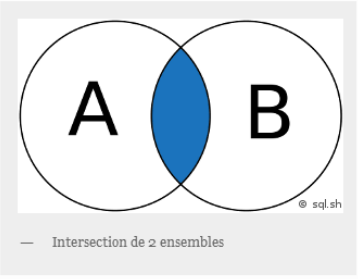
`SELECT * FROM LesGardiens, LesEmployes;`	Syntaxe déclarative.
`SELECT * FROM LesGardiens CROSS JOIN LesEmployes;`	Syntaxe algébrique.

Le résultat est bien évidemment le même dans les deux cas : 9 enregistrements dans la table gardiens et 6 dans la table employés nous donne bien 54 enregistrements dans le résultat.

	noCage	nomE	nomE	adresse
1	12	Berrut	Peyrin	Noumea
2	12	Berrut	Berrut	Sartene
3	12	Berrut	Maraninchi	Calvi

```
Result: 54 enregistrements ramenés en 24ms
At line 1:
SELECT * FROM LesGardiens, LesEmployes;
```

Jointure interne



La jointure interne sur les deux tables A et B permet de ne conserver que les enregistrements de A et B qui ont une valeur commune.

Donner les noms des animaux originaires du Kenya et qui ont contracté une grippe.

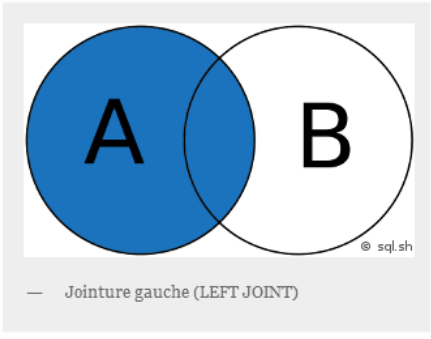
`SELECT A.nomA FROM LesAnimaux A, LesMaladies M`
`WHERE ((A.pays='Kenya') AND`
`(A.nomA=M.nomA) AND`
`(M.nomM='grippe')`
`);`
	Syntaxe déclarative.

`SELECT A.nomA FROM LesAnimaux A`
`JOIN LesMaladies M`
`ON ((A.pays='Kenya') AND`
`(A.nomA=M.nomA) AND`
`(M.nomM='grippe')`
`);`
	Syntaxe algébrique.

Les résultats :

	nomA
1	Charly
2	Charlotte

Les jointures externes : gauche



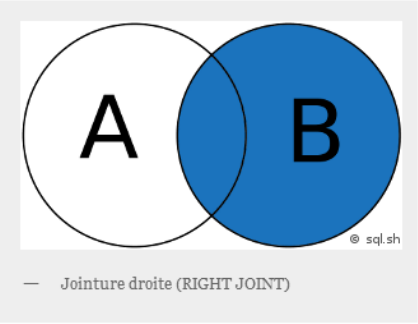
Permet de lister tous les résultats de la table de gauche même si il n'y a pas de correspondance dans la deuxième table. Le champ sera alors noté NULL.

Lister tous les animaux avec leur éventuelles maladies.

```
SELECT A.nomA, M.nomM
FROM LesAnimaux A
LEFT OUTER JOIN LesMaladies M ON A.nomA = M.nomA;
```

	nomA	nomM
1	Arthur	*NULL*
2	Charlotte	grippe
3	Charly	grippe
4	Charly	rage de dents
5	Chloé	grippe

Les jointures externes : droite



Permet de lister tous les résultats de la table de droite même si il n'y a pas de correspondance dans la première table. Le champ sera alors noté NULL.

Donner le liste de toutes les cages et de leurs gardiens éventuels, NULL si il n'y a pas de gardien affecté à la cage.

```
SELECT G.nomE, C.noCage
FROM LesGardiens G
RIGHT OUTER JOIN LesCages C ON C.noCage = G.noCage;
```

Non supporté par sqlite3.

Les non appartenances

On peut également sélectionner par une non appartenance à une liste. Exemple :

Donner les numéros et fonction des cages qui sont inoccupées.

```
SELECT C.noCage, C.fonction FROM LesCages C
WHERE C.noCage NOT IN (SELECT A.noCage FROM LesAnimaux A);
```

	noCage	fonction
1	2	aquarium
2	4	grand aquarium

4 Quelques ressources

4.1 Un site sur le langage Python et ses principales bibliothèques

http://www.python-simple.com/python-autres-modules-non-standards/sqlite3.php

4.2 Une ressource sur le langage SQL

https://sql.sh/

SQL.sh
Cours et tutoriels sur le langage SQL

5 Exercices TP base de données Zoo

Nom :	Note : / 20
	Classe :

Utiliser SQLiteDatabaseBrowser et Python pour les requêtes. testzoo_requete.py

Q1. Donner le schéma de la base zoo à partir du fichier zoo_exo.sql

Q2. Dessiner un schéma possible d'implantation du zoo : les allées et les cages respectant les données présentes dans la base.

Q3. Donner les noms des animaux qui n'ont pas été malade.

Q4. Donner les noms des gardiens qui s'occupent des lions.

Q5. Combien y-a-t'il d'animaux 'male' dans le zoo ?

Q6. Combien y-a-t'il d'allées dans le zoo ?

Q7. Donner l'année de naissance de l'animal le plus vieux.

Q8. Donner le nom de l'animal le plus vieux.

Q9. Donner les noms, type et pays d'origine des animaux qui partagent la cage de Milou.

Q10. Donner, pour chaque animal mâle, l'ensemble des maladies contractées (ensemble des couples nom d'animal, nom de maladie).

Q11. Donner les noms et type des animaux dont Peyrin est responsable.

Q12. Donner les requêtes permettant de remettre Taï Lung dans la bonne cage.

Q13. Intégrer dans la base un nouveau gardien d'origine Grenobloise pour les cages 2 et 4.