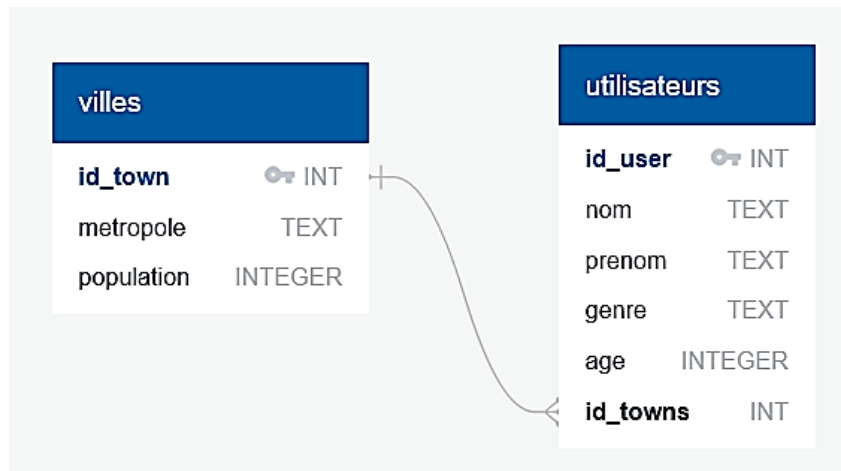


# Tp- Premières requêtes Sql sur Db Browser

Un fournisseur internet a besoin de connaître la localisation de ses clients. Il établit une base de données nommée *freeBranly*. Cette bdd contient les 2 relations définies sur le diagramme relationnel suivant :



L'objectif de ce travail est d'écrire vos premières requêtes Sql en utilisant le logiciel *DB Browser*. Il s'agit d'un outil graphique qui permet de gérer les bases de données SQLite. On l'utilisera pour créer la bdd *freeBranly*, en **exécutant uniquement des requêtes Sql**.

L'DB Browser accepte les requêtes « brutes » et en donne le résultat sous forme de tableaux, ce qui permet de bien les comprendre.

## 1- INSTALLATION DE DB BROWSER :

⇒ Télécharger la version **portable** du logiciel, à partir de la page *Downloads* du site <https://sqlitebrowser.org/> et copier ce fichier *.exe* dans votre dossier personnel **sur U** :

### Windows PortableApp

There is a PortableApp available, but it's still the previous (3.12.2) release version. It should be updated to 3.13.1 over the next few days:

- [DB Browser for SQLite - PortableApp](#)

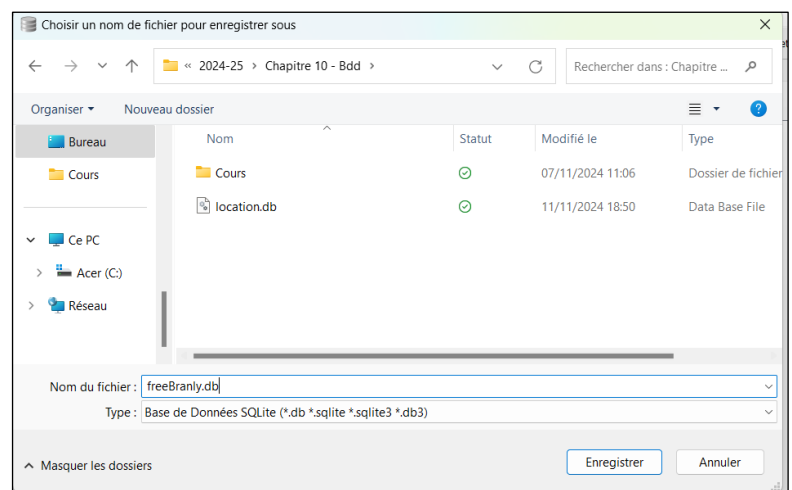
⇒ Lancer cet exécutable qui installera *Db Browser portable* dans ce dossier **sur U** .

## 2- CREATION D'UNE NOUVELLE BDD :

⇒ Ouvrir *DB Browser*



⇒ Utiliser l'onglet « *Nouvelle Base de Donnée* » pour en créer une. Vous la nommerez *freeBranly.db* .

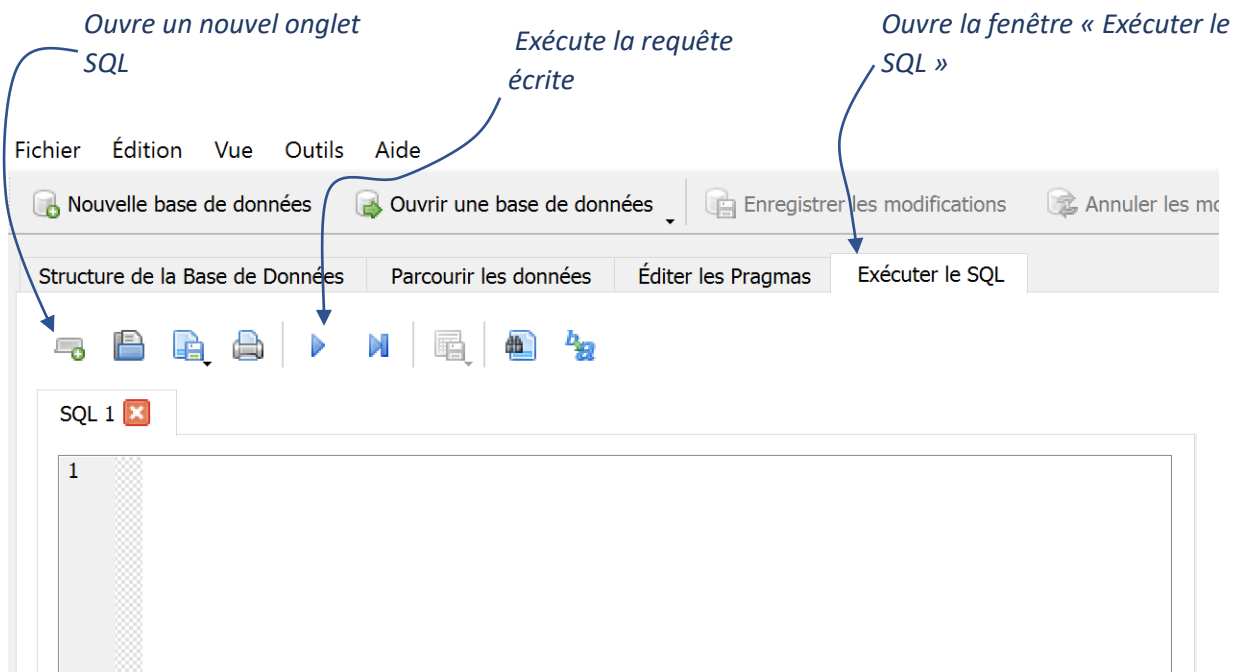
*DB Browser* vous propose ensuite de créer une première table : **fermez** la fenêtre correspondante, le reste du tp se fera uniquement à partir de requêtes Sql.



### 3- CREATION DES TABLES VILLES ET UTILISATEURS :

On va créer dans cette partie les tables *villes* et *utilisateurs* de la bdd freeBranly, en exécutant des commandes SQL . Pour saisir ces requêtes :

⇒ Cliquer sur l'onglet « Exécuter le SQL » pour ouvrir une fenêtre de saisie Sql. Pour exécuter une requête écrite dans cette fenêtre, il faudra simplement cliquer sur l'icone  . Pour la requête suivante, ne pas effacer la précédente, **ouvrir simplement une nouvelle fenêtre** en cliquant sur  .



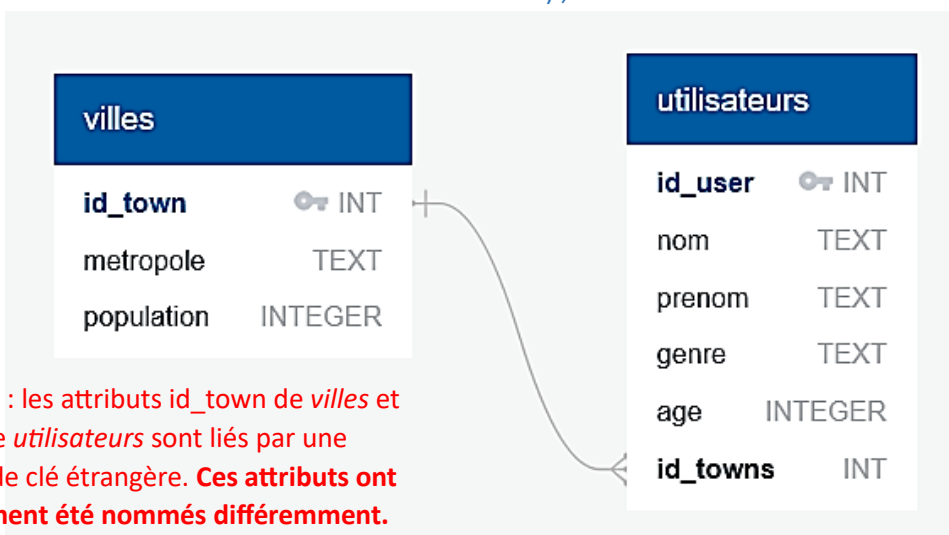
⇒ En utilisant une requête SQL à chaque fois (requête 1, puis requête 2), créer les tables *utilisateurs* et *villes*, définies ci-dessous :

*villes* ( id\_town INTEGER , metropole TEXT , population INTEGER )

*utilisateurs* ( id\_user INTEGER , nom TEXT, prenom TEXT, genre TEXT, age INTEGER, #id\_towns INTEGER)

On donne pour exemple, le début de la requête 1:

```
CREATE TABLE villes (  
id_town INTEGER PRIMARY KEY,  
metropole TEXT,  
.....  
);
```



**ATTENTION** : les attributs *id\_town* de *villes* et *id\_towns* de *utilisateurs* sont liés par une contrainte de clé étrangère. Ces attributs ont volontairement été nommés différemment.

⇒ Pour le compte-rendu de tp, écrire toutes les requêtes SQL demandées, dans un fichier SQL ouvert avec *VisualStudioCode* et enregistré avec le nom *freeBranly.sql*.

Vous écrirez le numéro de la requête en utilisant un commentaire (double tiret pour le Sql). Vous uploaderez ce fichier SQL en fin de Tp, sur *nsibranly.fr*. Par exemple, le début de ce fichier incomplet est donné ci-contre : →

```
-- Requete 1 :
CREATE TABLE villes (
    id_town INTEGER PRIMARY KEY,
    metropole TEXT,
    population INTEGER
);

CREATE TABLE utilisateurs (
    id_user INTEGER ,

    id_towns INTEGER,
    PRIMARY KEY(id_user),
    FOREIGN KEY(id_towns) REFERENCES villes(id_town)
);
```

#### Requête 1.

```
CREATE TABLE villes (
    id_town INTEGER PRIMARY KEY,
    metropole TEXT,
    population INTEGER
);
```

#### Requête 2.

```
CREATE TABLE utilisateurs (
    id_user INTEGER PRIMARY KEY ,
    nom TEXT,
    prenom TEXT,
    genre TEXT,
    age INTEGER,
    id_towns INTEGER,
    FOREIGN KEY(id_towns) REFERENCES villes(id_town)
);
```

### 4- REPLISSAGE DES TABLES :

⇒ Requête pour insérer les enregistrements suivants dans *villes* :

id_town	metropole	population
Filtre	Filtre	Filtre
1	LYON	513000
2	PARIS	2161000
3	MARSEILLE	861000
4	MACON	33000

#### Requête 3.

```
INSERT INTO villes VALUES
(1, 'LYON', 513000),
(2, 'PARIS', 2161000),
(3, 'MARSEILLE', 861000),
(4, 'MACON', 33000);
```

⇒ Requête pour insérer les enregistrements suivants dans *utilisateurs* :

id_user	nom	prenom	genre	age	id_towns
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	Chouhan	Jean	homme	50	1
2	Durand	Louis	homme	37	1
3	GranJean	Alice	femme	45	2
4	Bobet	Louison	homme	27	3

Requête 4.

```
INSERT INTO utilisateurs(id_user,nom,prenom,genre,age,id_towns) VALUES
(1, 'Chouhan', 'Jean', 'homme', 50, 1),
(2, 'Durand', 'Louis', 'homme', 37, 1),
(3, 'GranJean', 'Alice', 'femme', 45, 2),
(4, 'Bobet', 'Louison', 'homme', 27, 3);
```

## 5- MODIFICATION DES TABLES :

- Ajouter un 5<sup>ème</sup> enregistrement à la table « utilisateur » :

5	Champin	Arnaud	homme	23	1
---	---------	--------	-------	----	---

Requête 5.

```
INSERT INTO utilisateurs VALUES
(5, 'Champin', 'Arnaud', 'homme', 23, 1);
```

- Modifier l'âge de Durand par 83 :

Requête 6.

```
UPDATE utilisateurs SET age = 83 WHERE nom = 'Durand';
```

## 6- LECTURE DANS LES TABLES, SANS JOINTURE :

⇒ On désire obtenir tous les attributs de tous les enregistrements de *villes* :

Requête 7.

```
SELECT * FROM villes;
```

	id_town	metropole	population
1	1	LYON	513000
2	2	PARIS	2161000
3	3	MARSEILLE	861000
4	4	MACON	33000

⇒ On désire obtenir le nom des villes dont la population est supérieure à 500 000 habitants :

Requête 8.

	metropole
1	LYON
2	PARIS
3	MARSEILLE

```
SELECT metropole FROM villes
WHERE population > 500000;
```

⇒ On désire obtenir le nom des villes qui ont un « R » dans ce nom :

Requête 9.

	metropole
1	PARIS
2	MARSEILLE

```
SELECT metropole FROM villes
WHERE metropole LIKE '%R%';
```

⇒ On désire obtenir le prénom et l'âge des utilisateurs de sexe masculin :

Requête 10.

	prenom	age
1	Jean	50
2	Louis	83
3	Louison	27
4	Arnaud	23

```
SELECT prenom , age FROM utilisateurs
WHERE genre = 'homme';
```

## 7- OPERATIONS D'AGREGATION :

- Calculer la somme de tous les âges et la ranger dans l'attribut « somme\_age » (utiliser un **alias**).

Requête 11.

	somme_age
1	228

```
SELECT SUM(age) AS somme_age FROM utilisateurs;
```

- Donner le nombre d'utilisateurs (utiliser un **alias**) :

Requête 12.

	nb
1	5

```
SELECT COUNT(*) AS nb FROM utilisateurs;
```

- Donner la moyenne des âges des utilisateurs et ranger la dans l'attribut « ageMoyen » :

Requête 13.

	ageMoyen
1	45.6

```
SELECT AVG(age) AS ageMoyen FROM utilisateurs;
```

- Donner l'âge maximum des utilisateurs :

	age_maxi
1	83

Requête 14.

```
SELECT MAX(age) AS age_maxi FROM utilisateurs
```

## 8- SUPPRESSION COMPLETE D'UNE TABLE :

Créer une nouvelle table nommée « *maTable* » dont le schéma relationnelle est *maTable ( id INTEGER )*, puis exécuter la requête DROP TABLE maTable qui supprime cette table de la bdd.

Requêtes 15. (2 requêtes : CREATE ... et DROP TABLE ....)

```
CREATE TABLE maTable (  
    id INTEGER  
);  
DROP TABLE maTable;
```

## 9- JOINTURE DES 2 TABLES :

⇒ On désire obtenir les noms et prénoms des clients qui habitent LYON.  
Ecrivez la requête correspondante.

Requête 16.

	nom	prenom
1	Chouhan	Jean
2	Durand	Louis
3	Champin	Arnaud

```
SELECT nom,prenom FROM utilisateurs  
JOIN villes ON utilisateurs.id_towns = villes.id_town  
WHERE metropole = 'LYON';
```

⇒ On désire obtenir les nom, prenom et nombre d'habitants de la ville dans laquelle vit Durand :

	nom	prenom	population
1	Durand	Louis	513000

Requête 17.

```
SELECT u.nom,u.prenom, v.population FROM villes AS v  
JOIN utilisateurs AS u ON v.id_town = u.id_towns  
WHERE u.nom = 'Durand'
```

⇒ On désire obtenir les nom, prénom et ville des utilisateurs qui vivent dans une ville de plus de 1 000 000 habitants :

	nom	prenom	metropole
1	GranJean	Alice	PARIS

Requête 18.

```
SELECT u.nom,u.prenom FROM utilisateurs AS u
JOIN villes AS v ON u.id_towns = v.id_town
WHERE v.population > 1000000
```

⇒ Ecrire une requête qui récupère tous les attributs de la table *villes* avec en plus les attributs des *utilisateurs*, pour tous les enregistrements de *villes* :

	id_town	metropole	population	id_user	nom	prenom	genre	age	id_towns
1	1	LYON	513000	5	Champin	Arnaud	homme	23	1
2	1	LYON	513000	1	Chouhan	Jean	homme	50	1
3	1	LYON	513000	2	Durand	Louis	homme	83	1
4	2	PARIS	2161000	3	GranJean	Alice	femme	45	2
5	3	MARSEILLE	861000	4	Bobet	Louison	homme	27	3
6	4	MACON	33000	NULL	NULL	NULL	NULL	NULL	NULL

Requête 19.

```
SELECT * FROM villes AS v
LEFT JOIN utilisateurs AS u ON v.id_town = u.id_towns
```

⇒ Ecrire une requête qui récupère tous les attributs de la table *villes* avec en plus les attributs des *utilisateurs*, pour seulement les enregistrements de *villes* répertoriés dans *utilisateurs* :

	id_town	metropole	population	id_user	nom	prenom	genre	age	id_towns
1	1	LYON	513000	1	Chouhan	Jean	homme	50	1
2	1	LYON	513000	2	Durand	Louis	homme	83	1
3	2	PARIS	2161000	3	GranJean	Alice	femme	45	2
4	3	MARSEILLE	861000	4	Bobet	Louison	homme	27	3
5	1	LYON	513000	5	Champin	Arnaud	homme	23	1

Requête 20.

```
SELECT * FROM villes AS v
JOIN utilisateurs AS u ON v.id_town = u.id_towns
```

## 10- SUPPRESSION D'UN SEUL ENREGISTREMENT SUR LA TABLE VILLES :

⇒ On désire supprimer l'enregistrement relatif à la ville de MACON dans la table *villes* :

Requête 21.

```
DELETE FROM villes WHERE metropole = 'MACON'
```

⇒ On désire supprimer l'enregistrement relatif à la ville de MARSEILLE dans la table *villes* :

## Requête 22.

```
DELETE FROM villes WHERE metropole = 'MARSEILLE'
```

Cette requête est-elle autorisée ? Expliquer ... (réponse dans le fichier Sql qui sera uploadé)

Cette requête n'est pas autorisée car un des enregistrements de la table *utilisateurs* fait référence à cet enregistrement. Pour pouvoir supprimer de la table *ville* l'enregistrement relatif à Marseille, il faudrait tout d'abord supprimer tous les enregistrements de la table *utilisateurs* qui font référence à cette ville de Marseille.

```
L'exécution s'est terminée avec des erreurs.  
Résultat : FOREIGN KEY constraint failed  
À la ligne 1 :  
DELETE FROM villes WHERE metropole = 'MARSEILLE'
```