BdD – 2 h (+hors séance)

TP-S3-ITC3

Gestion d'une base de données : 1 - création de la base

Objet de l'étude

Nous allons étudier les systèmes de gestion de bases de données (SGBD) de type relationnel au travers d'une étude qui se découpe en deux parties.

Dans cette première partie, une base de données sera construite à l'aide de plusieurs outils découverts pour l'occasion. Ce sera l'occasion de rappeler le vocabulaire et le fondement des bases de données en s'appuyant sur les ressources de cours ou disponibles sur l'ENT Moodle.

Outils utilisés

- De quoi prendre des notes personnelles : voir les détails dans les sections suivantes.
- Un ordinateur personnel (de bureau) disponible au lycée où à la maison.
- Logiciel de gestion de base de données SQLiteBrowser.
- Tableur Excel ou Libre Office.
- Éditeur de texte Notepad ou équivalent.

Documents utilisés (disponibles sur Moodle)

- Diaporama « Éléments d'initiation aux bases de données et au langage d'interrogation SQL » les diapositives utiles auxquelles se référer sont repérées **[Dnn]** dans le texte de ce TP.
- Fiche technique « SQLite pour la gestion des bases de données » (aperçu et installation condensés)

Notations des actions souris

Parfois, pour éviter d'alourdir la description des manipulations dans le texte du sujet, nous adopterons les conventions suivantes :

- **CL** : clic gauche avec la souris (choix ou validation) ;
- **CLDR** : clic droit avec la souris (menu contextuel) ;
- **DBCL** : double clic avec le bouton gauche.

Organisation

- Prendre des notes personnelles qui rappellent la démarche et appuyées par les indications.
- Conserver une copie des documents fournis pour ce TP (via Moodle) et des documents numériques qui auront été produits durant la séance (fichiers personnels, scripts, etc.).
- Conserver les notes informatiques produites (éditeur, documents textes, documents graphiques, etc.), en les enregistrant, soit dans l'espace personnel de stockage des documents sur Moodle (à préférer), ou alors dans un dossier de votre espace de stockage personnel (souvent c'est le lecteur H) associé à votre compte utilisateur au lycée (dénommé « compte Kwartz »).
- Dupliquez aussi toutes ses productions sur une clef de stockage (ou sur un espace de stockage en ligne fiable) pour un travail personnel hors séance ou en cas d'impossibilité d'accéder aux données du serveur. Il est important de se familiariser avec cette pratique de la sauvegarde de ses données.
- Poursuivre le travail à son rythme en dehors de la séance.

Compte rendu

Garder l'ensemble de vos notes manuscrites avec le sujet du TD/TP même si un compte rendu de vos travaux n'est pas demandé : remarques personnelles et apports présentés au tableau pour des usages ultérieurs ou dans d'autres domaines d'application.

I. Introduction

Pour construire et gérer une base de données (BdD), l'usage des logiciels de gestion tels que MySQL ou PostgreSQL est courant en « production » (pour construire les applications web en en général).

Pour cette étude, nous allons utiliser une autre méthode, **SQLite** [WIKI1] et **DB Browser for SQLite**, qui offrent un environnement pratique, plutôt efficace, mais surtout dont la prise en mains est aisée et rapide ; il est de surcroît plutôt « visuel » pour créer, gérer et interroger notre base de données.

Il est aussi possible de gérer les BdD avec des modules Python qui permettent d'accéder et de gérer des BdD : ce moyen ne fait pas l'objet de manipulation dans l'étude proposée.

II. Préparation des connaissances et des outils

SQLite est un moteur de base de données relationnelle, c'est à dire qu'il peut être inséré dans une application pour assurer la gestion de données sans avoir à concevoir toutes les fonctionnalités de leur gestion. Comme son nom le rappelle, ce logiciel met en œuvre des bases de données gérées avec le langage **SQL** [WIKI2] pour *Structured Query Language*, ou, en français, langage de requête structurée. C'est un langage informatique normalisé servant à créer et exploiter des bases de données relationnelles. La partie langage de SQL permet de créer et modifier des bases de données pour manipuler les données, essentiellement pour en ajouter, les modifier ou supprimer, ainsi que les rechercher pour les extraire de la base. Ces opérations peuvent être effectuées en mode console, c'est à dire en saisissant le texte des instructions SQL dans un interpréteur de commandes.

Le logiciel **DB Browser for SQLite** est disponible en ligne. Les indications pour l'installer font l'objet de la fiche technique « SQLite pour la gestion des bases de données » évoquée en première page.

Q.1 Étudier les diapositives du cours (**[D3]** à **[D10]**, ou **[D3..10]**) afin de se rappeler le vocabulaire des bases de données.

Q.2 Si ce n'est déjà fait sur son poste de travail, installer l'application « DB Browser for SQLite » (DBB par la suite) en se référant à sa fiche technique.

III. Début de la construction de la base de données

Nous allons créer une base de données de gestion des étudiants dont le nom sera « **gestion_etudiants** ». L'extension (**.db**) n'est pas à indiquer, le logiciel se chargeant de l'ajouter. Un guide est donné dans le fichier de suivi de construction : « 2024ITC3_TP1_Suivi-creation-BdD.pdf ». Il présente les vues des fenêtres du logiciel au fur et à mesure de la construction.

Création de la base

Q.3 Lancer (**DBCL**) le logiciel DBB. Créer la base de données « gestion_etudiants » par un **CL** sur le bouton « Nouvelle base de données » (« *New Database* » si c'est une version anglaise). La version française étant disponible facilement, il ne sera plus fait mention de la version anglaise.

Quelques rappels pratiques. À l'instar des pratiques en Python, pour éviter d'éventuels problèmes de codage, utiliser l'écriture la plus internationale possible : n'utiliser que des caractères non accentués (pas de à, ê, etc.), éviter les caractères spéciaux (#, -, %, etc.) ou des espaces dans le nom des bases, des tables et des attributs. Utiliser aussi un nommage informatif des entités créées par exemple, préférer « etudiant » à « etu ».

Q.4 Enregistrer et ranger ce fichier dans son dossier de travail. Penser aussi à enregistrer régulièrement ses travaux (signe de bonnes pratiques, utiliser les raccourcis clavier).

Création de la première table

Dès que la base est sauvegardée, elle est active. Puisqu'elle est nouvelle, elle est vide. La création d'une première table est immédiatement proposée avec l'apparition de la fenêtre « Éditer la définition de la table ». Celle-ci se décompose en deux parties principales : l'édition avec l'éditeur graphique dans le bloc supérieur tandis que les commande correspondantes en langage SQL apparaissent dans celui du bas, ici « `CREATE TABLE '' ();` ». Cette organisation est quasi systématique.

Q.5 Suivre les étapes successives suivantes pour créer la table « **lycees** » :

- Nommer la table « lycees » dans le champ en haut de la fenêtre. Le pluriel est retenu pour les tables afin de signifier qu'il y a plusieurs n-uplets de données. L'affichage de la requête s'adapte simultanément ;
- La partie supérieure des champs (champ=attribut) contient un champ « Field1 » et la requête SQL « CREATE… » apparaît en dessous ;
- Il est possible d'effacer un champ et le déplacer pour les réorganiser. Ici, nous avons :
  - Renommer l'attribut créé par un **DBCL** sur « Field1 » et remplacer par « id_lycee » (colonne « Nom ») ;
  - Choisir ou conserver le type « INTEGER » dans la colonne « Type » ;
  - Cet attribut est la clef primaire de la table qui doit être unique (pas de doublon dans ses données) : cocher « cp » (clé primaire) afin de l'indiquer. Pour éviter la saisie de deux clefs identiques, cocher « IA » (autoincrémentation) pour incrémenter automatiquement sans action de l'usager et « U » (unique). Cocher « not null » pour forcer le remplissage de l'attribut avec une donnée (pour ne pas le laisser vide) ;
  - Ne rien indiquer dans la colonne « Default ».

À l'issue de ces étapes, la fenêtre ressemble à la Figure 1.

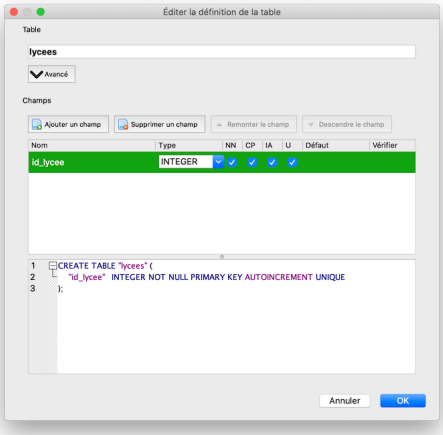

Figure 1 – Fenêtre de création de table et création d'un attribut.

S'intéresser au cadre inférieur qui présente la requête SQL qui résume la création opérée avec les paramètres choisis pour les colonnes. C'est la première requête SQL, ici pour créer la table. Ce schéma est similaire pour la suite avec d'autres requêtes.

Q.6 Terminer la création des attributs en effectuant un **CL** du bouton « Ajouter un champ » avec les définitions et paramètres suivants :

- Attribut « nom » : type TEXT, NOT NULL ;
- Attribut « ville » : type TEXT, NOT NULL

Q.7 Valider par un **CL** sur le bouton « Ok ». La fenêtre se ferme et la table apparaît dans le bloc en dessous de l'onglet « Structure de la Base de Données » de la fenêtre principale. Les autres parties sont aussi mises à jour comme l'indique la Figure 2.

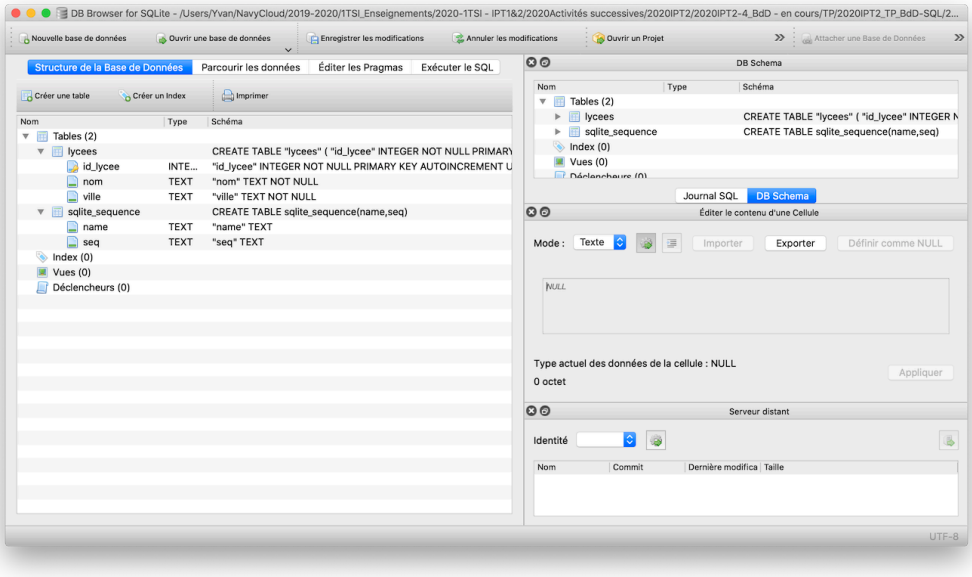

Figure 2 – Fenêtre de création de base de données.

Cette fenêtre peut être complétée alors que le schéma (structure) apparaît en haut à droite (Figure 2) en donnant le code des requêtes SQL de création de la table. L'onglet de gauche (Journal SQL) précise l'historique des opérations depuis l'ouverture du logiciel : c'est un fichier « log », ou « journal » en français (Figure 3).

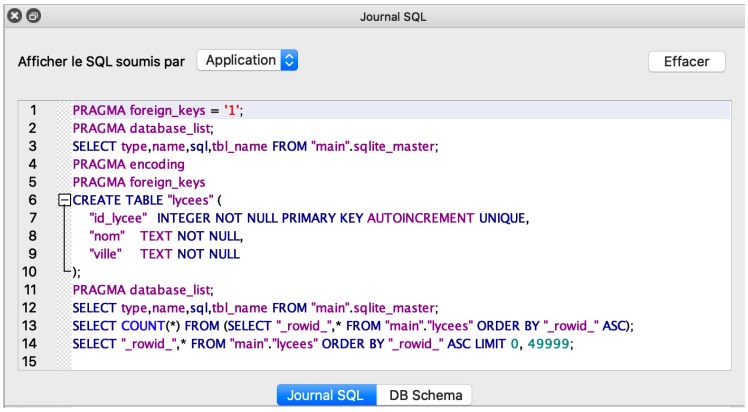

Figure 3 – Cadre de droite de la fenêtre de création de base de données.

Cette partie a permis de décrire en détail la création d'une base de données et de sa première table « **lycees** ». D'autres tables vont s'ajouter à celle-ci : leur construction fait l'objet d'une nouvelle partie.

IV. Finalisation de structure de la base de données : les autres tables

IV.1. Table « classes »

À la différence de la création précédente, la table « **classe** » peut être créée avec une requête SQL. Attention à l'indentation des blocs et à l'ordre des mots-clefs qui est à respecter. Ils sont par convention toujours écrits en **`LETTRES CAPITALES`**, alors que **les noms d'attributs sont écrits en lettres minuscules** délimités par des apostrophes (") :

```
CREATE TABLE "classes" (
      "id_classe" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
      "nom" TEXT NOT NULL,
      "niveau" INTEGER NOT NULL
);
```

Manipulations

Q.8 Ouvrir la fenêtre d'édition après un **CL** sur le bouton de l'onglet « Exécuter le SQL » (Figure 4). Saisir la requête précédente (ou effectuer un « copier/coller », rappel **CTRL-C/CTRL-V**) dans la zone supérieure de la fenêtre.

Q.9 Contrôler la syntaxe puis lancer la requête par un **CL** sur le bouton « play » de la barre d'icônes ou presser « F5 » (d'autres raccourcis dans la bulle d'aide au survol de l'icône).

Si aucune erreur n'apparaît, la table est créée. Un message l'indique dans le cadre inférieur : « `Result: Requête exécutée avec succès. A pris 0 ms…` ».

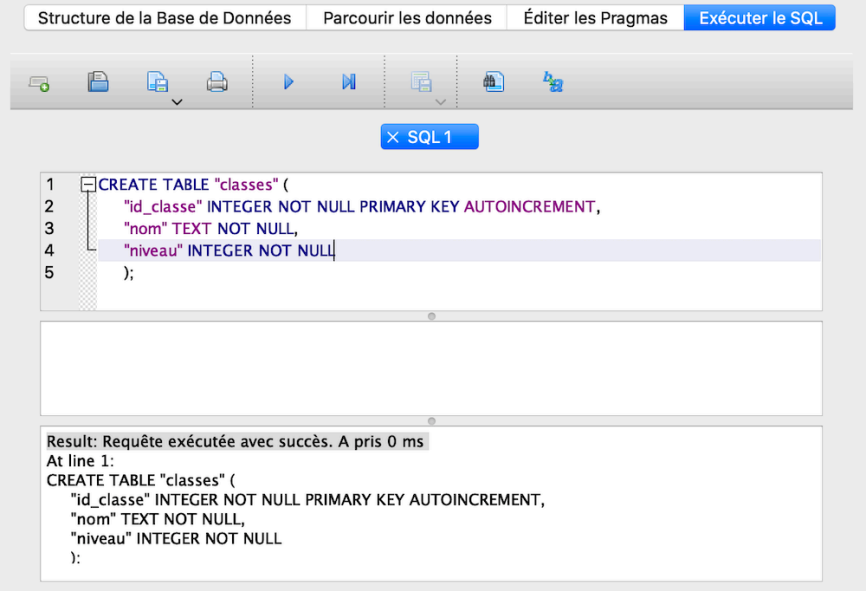

Figure 4 – Onglet et fenêtres d'exécution des requêtes SQL.

Q.10 Vérifier le résultat après un **CL** sur le bouton d'onglet « Structure de la Base de Données ».

IV.2. Table « etudiants »

Q.11 Avec l'assistant ou en construisant la requête, au choix, créer la table « **etudiants** » disposant des attributs suivants :

- `id_etudiant : INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,`
- `nom : TEXT NOT NULL,`
- `prenom : TEXT NOT NULL,`
- `email : TEXT NULL,`
- `sexe : INTEGER DEFAULT '0' NOT NULL,`
- `adresse : TEXT NULL,`
- `telephone : TEXT NULL,`
- `id_lycee : INTEGER DEFAUT '1' NULL,`
- `note_moyenne_bac : NUMERIC NULL`

Remarques

- `sexe` a comme valeur par défaut `0` qui correspond au sexe masculin. Cette valeur est choisie par défaut car il y a plus de garçons que de filles dans les filières d'ingénieurs (cette situation reste à déplorer) ;
- `telephone` est de type `TEXT` car ce serait une erreur de le stocker sous forme de nombre (les opérations ne seraient pas possibles avec ce nombre) ;
- `id_lycee` correspond au lycée de terminale avec comme valeur par défaut `1` qui signifiera « lycée inconnu » ;
- `note_moyenne_bac` est de type `NUMERIC` (et pas un entier, type `INTEGER`).

IV.3. Table « concours »

Q.12 Avec sa méthode préférée (parmi celle étudiées), créer la table « **concours** » avec :

- `id_concours : INTEGER PRIMARY KEY NOT NULL,`
- `nom : TEXT NOT NULL`

IV.4. Table « ecoles »

Q.13 Créer la table « **ecoles** » avec :

- `id_ecole : INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,`
- `acronyme : TEXT NOT NULL,`
- `nom : TEXT,`
- `ville : TEXT NOT NULL`

IV.5. Tables issues d'associations

Les associations, ou jointures **[D17]**, permettent de créer des tables plus complexes que celles créées manuellement auparavant car elles font intervenir des clefs étrangères (*foreign key*). **[D11..13]**

Les **clefs étrangères** identifient un attribut (colonne) ou un ensemble d'attributs d'une table pour référencer une colonne ou un ensemble de colonnes d'une autre table qui constitue la table référencée.

Entre les tables « etudiants » et « classes »

Voici la requête SQL pour créer la table « **etudiant_classe** » :

```
1  CREATE TABLE "etudiant_classe" (
2       "id_etudiant" INTEGER NOT NULL,
3       "id_classe" INTEGER NOT NULL,
4       "annee" INTEGER NOT NULL,
5       "boursier" INTEGER DEFAULT 0,
6       PRIMARY KEY(id_etudiant, id_classe, annee),
7       FOREIGN KEY(id_classe) REFERENCES classes
8       ON DELETE CASCADE
9       ON UPDATE CASCADE
10      FOREIGN KEY(id_etudiant) REFERENCES etudiants
11      ON DELETE CASCADE
12      ON UPDATE CASCADE
13 );
```

Q.14 En s'aidant de la notice https://sql.sh ou https://www.mysqltutorial.org , expliquer à l'écrit les lignes suivantes (utiliser le moteur de recherche interne à ces sites) :

- « `foreign Key` » ;
- « `primary key` » ;
- « `on delete` » ;
- et « `on cascade` ».

Entre les tables « etudiants » et « concours »

```
1  CREATE TABLE "note_concours" (
2       "id_concours" INTEGER NOT NULL,
3       "id_etudiant" INTEGER NOT NULL,
4       "moyenne" NUMERIC(4,2),
5       "annee" INTEGER NOT NULL,
6       "mois" INTEGER NOT NULL,
7       PRIMARY KEY(id_concours,id_etudiant,annee,mois),
8       FOREIGN KEY(id_concours) REFERENCES concours
9       ON UPDATE CASCADE
10      ON DELETE SET NULL
11      FOREIGN KEY(id_etudiant) REFERENCES etudiant
12      ON UPDATE CASCADE
13      ON DELETE CASCADE
14 );
```

Entre les tables « etudiants », « concours » et « ecoles »

```
CREATE TABLE "poursuites" (
      "id_etudiant" INTEGER NOT NULL ,
      "id_ecole" INTEGER NOT NULL,
      "id_concours" INTEGER NOT NULL,
      "annee" INTEGER NOT NULL,
      PRIMARY KEY(id_etudiant,id_ecole),
      FOREIGN KEY(id_etudiant) REFERENCES etudiants
      ON UPDATE CASCADE
      ON DELETE CASCADE,
      FOREIGN KEY(id_ecole) REFERENCES ecoles
      ON UPDATE CASCADE
      ON DELETE SET NULL,
      FOREIGN KEY(id_concours) REFERENCES concours
      ON UPDATE CASCADE
      ON DELETE SET NULL
);
```

Maintenant que la base de données est créée, des données vont pouvoir lui être ajoutées. C'est le but après tout, gérer des données…

V. Contenu de la base

V.1. Insertion des données

Manipulations : insertion manuelle avec l'interface graphique

Q.15 Suivre les indications données ci-dessous.

Pour accéder aux données, sélectionner l'onglet « Parcourir les données » par **CL**, puis choisir la table « classes » par **CL**.

Insérer un n-uplet (ou enregistrement), **CL** sur « Nouvel enregistrement », avec :

- nom : TSI1, niveau : 1

Recommencer pour les autres n-uplet :

- TSI2,2 ;
- PCSI,1 ;
- MP,2 ;

Remarques

En cas d'erreur, un **DBCL** permet de modifier la cellule concernée.

En cas d'excès de n-uplets : se placer sur le n-uplet à éliminer et **CL** sur « Effacer enregistrement ».

La valeur de « id_classe » n'est pas saisie car cet attribut est en AUTOINCREMENT.

Cette manière d'insérer des données semble simple et pratique mais elle devient très vite pénible lorsque la quantité de données à ajouter est importante. Il est alors possible d'utiliser des requêtes (pré enregistrées) ou une importation depuis un fichier de sauvegarde.

Importation avec une requête « insert »

Q.16 Dans l'onglet des requêtes SQL, écrire la requête `INSERT` (utiliser la documentation et/ou le formulaire SQL) permettant l'ajout des classes :

- PTSI, 1 ;
- PSI, 2 ;
- ATS, 2.

Pour saisir de grandes quantités de données ou restaurer une base détériorée (plantée, crashée ou autre), il est possible d'importer des données se trouvant dans un fichier « dump », (ou vidage).

Par importation

Ici, nous allons compléter les données à partir de sources de données extérieures enregistrées dans des fichiers au format « csv » (*comma-separated values*) comportant l'extension « csv » dans le dossier TP1 sur Moodle.

Q.17 Télécharger les fichiers de données (csv) de Moodle (archive zip à décompresser) et les stocker dans le dossier de travail.

Q.18 Ouvrir le fichier « concours.csv » avec un éditeur de textes (notepad ou wordpad par exemple) et vérifier le contenu. La première ligne contient le nom de chacun des champs séparés par un caractère de séparation (virgule, point-virgule ou autre). Ces noms doivent correspondre exactement aux noms des attributs de la table d'affectation.

Q.19 Suivre les indications suivantes :

- Dans l'onglet « Parcourir les données », choisir « concours » comme table de destination ;
- Constater que la table « concours » ne contient pas de données ;
- Menu Fichier>Importer>Table depuis une fichier CSV ; dans la fenêtre qui apparaît localiser le fichier « concours.csv », **CL** sur « ouvrir » et importer.
- Observer le contenu, puis répondre au paramétrage de format :
  - Taper le nom de la table (c'est plutôt une sécurité) ;
  - Cocher « *Nom de colonne en première ligne* » pour que les noms de champ ne soit pas considérés comme des données ;
  - Préciser le caractère de séparation (*séparateur de champ*) observé dans l'éditeur ;
  - Préciser le caractère d'apostrophe pour les blocs de données ;
  - Ne pas changer le codage (UTF-8)
  - **CL** sur « OK » pour importer les données avec message d'alerte de confirmation ;
  - Corriger le problème rencontré et achever l'importation ;
  - Un message d'information apparaît quand l'importation est terminée.

V.2. Ajout de données simples

Q.20 Refaire une importation pour remplir la table « lycees » à partir du fichier « lycees.csv ».

Q.21 Avec un tableur (Excel ou Libre Office Calc), créer un fichier de type « csv » comme celui utilisé pour « concours » et pour « `lycees` ». Il sera nommé : « ecoles.csv ».

Il comportera 4 colonnes : « `id_ecole` », « `acronyme` », « `nom` » et « `ville` ».

Q.22 Ajouter une dizaine d'école à cette feuille, sauvegarder au format « `csv` », puis importer les données dans la table « `ecoles` ».

V.3. Ajout de données complexes

Nous souhaitons maintenant ajouter un étudiant. Mais il est nécessaire de connaître l'identifiant de son lycée, voire, le cas échéant d'ajouter, son lycée dans la table « `lycees` ».

Manipulations

Q.23 Écrire la requête SQL pour ajouter l'étudiant « Zinedine Zidane » venant du lycée « Thiers » de « Marseille ». Il sera nécessaire d'utiliser une sous requête pour trouver l'`id_lycee`.

Importation

Q.24 Utiliser le fichier « `etudiants`.csv » en remplacement du contenu actuel de la table « etudiants ».

Comme il y a une limite à ce que l'on peut faire « à la main », cela devient vite très difficile et pénible d'ajouter de cette manière un étudiant dans une classe dans la table « etudiant_classe » car il faut connaître l'id_lycee qui est stocké dans la table « lycees ».

Manipulations

Q.25 Importer le fichier « etudiant_classe.csv » pour incorporer les étudiants dans des classes. Pour cela, nous supposons que les étudiants et les classes ont précédemment été créés.

L'insertion de données dans la table « poursuites » ne peut pas se faire « à la main » car elle relie les trois tables : « `etudiants` », « `concours` » et « `ecoles` ».

C'est pourquoi, nous utiliserons par la suite l'application « `gestion_etudiants` » pour simplifier cette saisie.

V.4. Application Gestion_etudiants

Manipulations

Suivre les étapes suivantes :

- Démarrer l'environnement Python ;
- CL sur Fichier puis Nouveau projet ;
- Si c'est la première fois, désigner un dossier pour les projets (choisir un dossier sur votre espace de stockage sur le réseau) ;
- Nommer le projet « Gestion_etudiants » ;
- Rendez-vous dans le dossier qui vient d'être créé pour le projet et y copier le contenu du dossier « logiciel_cpge » (une trentaine de fichiers) ;
- Renommez votre fichier de base de donnée en « GestionEtudiants.sqlite » ;
- Copiez ensuite votre fichier renommé dans le dossier « Gestion_etudiants » du projet ;
- Dans l'environnement, ouvrir le fichier « main.py » (en DBCL dessus) puis exécuter (touche F5).

Manipulations

Le logiciel vous permet :

- D'ajouter de nouveaux étudiants ;
- De les faire évoluer vers de nouvelles classes ;
- De leur faire passer des concours ;
- De leur faire intégrer des écoles.

VI. Références

- [SQL] – **Cours SQL**, http://sql.sh/cours/.
- [WIKI1] – **SQLite**, http://fr.wikipedia.org/wiki/SQLite.
- [WIKI2] – **SQL**, http://fr.wikipedia.org/wiki/Structured_Query_Language.