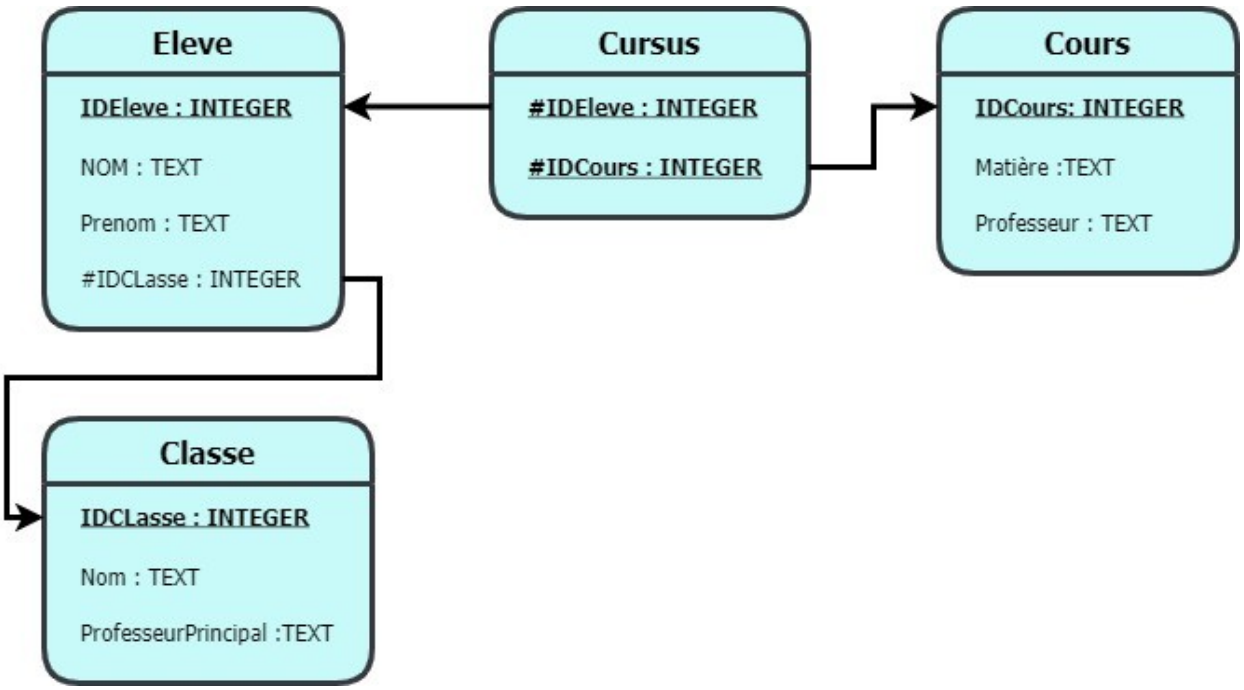
Mini projet 4 : base de données

Récupérez les fichiers :
base_mp.db qui contient la base sqlite (ce format n'est pas directement lisible).
base_mp.sql qui contient les instructions SQL pour initialiser la base. (c'est un fichier texte)
base_mp.py qui constitue un exemple d'utilisation de Python pour interagir avec la base sqlite.

Vous allez travailler sur une base dont la représentation graphique du schéma relationnel est :



Étape 1 : prise en main

Les deux premières tables sont déjà implémentées et pour cette étape on n'utilisera que celles-ci.

1) En vous inspirant de la fonction **select_toutes_classes(conn)**, créez une fonction **select_tous_eleves(conn)** qui renvoie la liste de tous les élèves.

2) Créez une fonction **eleves_prenom(conn, nom)** qui renvoie la liste de tous les prénoms des élèves dont le nom est **nom**.

3) Créez une fonction **eleve_in(conn, nom, prenom)** qui vérifie s'il y a un élève nommé **nom prenom** dans l'établissement, et qui renvoie le booléen correspondant.

4) Créez une fonction **classe_eleve(conn, nom, prenom)** qui renvoie le nom de la classe de l'élève nommé **nom prenom**.

5) Créez une fonction **nb_eleves_classe(conn, nom_classe)** qui renvoie le nombre d'élèves dans la classe **nom_classe**.

6) Complétez le menu console de la fonction **main()** de **base_mp.py** avec les 5 fonctionnalités que vous venez de définir.

Étape 2 : compléter la base

1) Ajoutez dans le fichier **base_mp.sql** la structure des tables Cours et Cursus à l'aide de la fonction **CREATE**. Dans ce même fichier, peuplez ces deux nouvelles tables de quelques enregistrements.

2) Créez une fonction **matiere(conn)** qui affiche toutes les matières enseignées (une seule fois pour chaque matière)

3) Créez une fonction **cursus_eleve(conn, nom, prenom)** qui affiche la liste de toutes les matières suivies par l'élève nommé **nom prenom**.

4) Créez une fonction **eleves_cours(conn, IDCours)** qui renvoie la liste des couples (nom, prenom) de tous les élèves suivant le cours dont l'identifiant est **IDCours**)

5) Créez une fonction **nb_eleves_cours((conn, IDCours)** qui renvoie le nombre d'élèves suivant le cours dont l'identifiant est **IDCours**.

6) Créez une fonction **test()** dans laquelle vous prévoirez des tests pour vérifier et illustrer que chacune des 4 fonctions précédentes fonctionne.
Pour chaque fonction, vous envisagerez les différentes situations possibles.
La fonction test devra afficher dans la console l'ensemble de la démarche.

Exemple de rendu console possible :

```
#####################################
Ensemble des tests de la fonction cursus_eleve
#####################################
Test fonction cursus_eleve pour Bourgeois Michel
NSI, Maths
#####
Test fonction cursus_eleve pour Brel Jacqueline
Cet élève ne fait pas partie de l'établissement.
#####
Test fonction cursus_eleve pour Marchal Dimitri
Cet élève fait partie de l'établissement, mais ne suit pas de cours.
#####
```

Étape 3 : perfectionnement (facultatif)

1) Vous redéfinirez le schéma relationnel en prenant en compte les remarques suivantes :
Les élèves et les professeurs sont des personnes. (Et oui !).
Chaque personne à un nom, un prénom.
Vous donnez votre nouveau schéma relationnel sous forme graphique.

2) Dans un nouveau répertoire au sein d'une nouvelle version (conservez bien à part les fichiers de étapes 1 et 2) vous redéfinirez le code sql et le code python pour s'adapter à votre nouvelle structure.
Vous répondrez aux mêmes question que celles des étapes 1 et 2.

3) Enfin, vous rajouterez une fonction **eleves_professeur(conn, nom, prenom)** qui renvoie la liste des couples (nom, prenom) des élèves ayant le professeur nommé **nom prenom** en cours.