DS 1

Informatique de tronc commun, classe de PC

Julien Reichert

Ce devoir consiste en une partie de programmation et deux parties sur des bases de données totalement indépendantes. Les fonctions et requêtes à écrire ne sont pas forcément de difficulté croissante.

Partie 1 : Programmation

Toutes les questions de cette section sont à traiter en Python et sont indépendantes.

Exercice 1.1 : Écrire une fonction prenant en argument deux entiers `n` et `k` et renvoyant une valeur correspondant à la formule $\sum_{i=1}^{n} i^k$ (à l'aide de boucles ou de récursions et sans formule directe).

Exercice 1.2 : Écrire une fonction prenant en argument une liste et renvoyant un booléen déterminant si un élément de la liste est présent en majorité absolue.

Exercice 1.3 : Écrire une fonction prenant en argument une liste de listes d'entiers et renvoyant une position du plus grand entier dans la liste de listes (on renverra le couple `(i, j)` ou de manière équivalente la liste de taille deux correspondante pour signaler que le maximum apparaît à l'indice `j` de la liste d'indice `i`). Si le maximum a plusieurs occurrences, n'importe laquelle peut être localisée.

Exercice 1.4 : Écrire une fonction prenant en argument un entier `n` entre zéro et un milliard et renvoyant le plus grand entier ayant les deux propriétés suivantes : (i) être inférieur ou égal à un milliard, et (ii) être égal au produit de `n` et d'une puissance de deux.

Exercice 1.5 : Écrire une fonction prenant en argument deux entiers naturels non nuls et renvoyant un booléen déterminant si les entiers sont premiers entre eux.

Partie 2 : Base de données « BD »

Ma collection de bandes dessinées n'a pas encore fait l'objet d'un stockage dans une base de données, mais si c'était le cas, la structure serait sans doute la suivante :

- `SERIE`, dont les attributs sont `id` (entier) et `titre` (chaîne de caractères) ;
- `COLLECTION`, dont les attributs sont `id_serie` (entier), `id_collection` (entier) et `titre_collection` (chaîne de caractères) ;
- `AUTEUR`, dont les attributs sont `id_serie` (entier), `nom_auteur` (chaîne de caractères) et `role` (chaîne de caractères) ;
- `ALBUM`, dont les attributs sont `id_collection` (entier), `numero` (entier), `titre` (chaîne de caractères), `date` (entier) et `possede` (booléen).

Quelques explications : une « collection » est une sous-série (identifiée par un entier unique et différent pour deux collections issues de séries différentes dans la base de données), souvent démarrée par un changement d'auteur, un auteur est associé à une série dès qu'il a contribué (quel que soit le rôle, qui peut être scénariste, dessinateur, coloriste…) à au moins un album, la date est limitée à une année et le booléen `possede` peut être vu comme valant `1` (si je dispose de l'album) ou `0` sinon.

Exercice 2.1 : Donner pour chaque table les clés et clés étrangères.

Exercice 2.2 : Justifier par le modèle entité-relation le schéma retenu pour la base de données.

Exercice 2.3 : Bien qu'on ait décidé de ne pas créer de table recensant des attributs de l'entité « auteur » (par exemple avec son état civil) pour simplifier, la table `AUTEUR` peut être considérée comme décrivant une relation. De quel type (1-1, 1-* ou *-*) est-elle ? Justifier.

Exercice 2.4 : Écrire une requête permettant d'obtenir le titre de la série d'identifiant `19`.

Exercice 2.5 : Écrire une requête permettant de déterminer l'auteur de la série `"Tintin"`.

Exercice 2.6 : Écrire une requête permettant de déterminer l'année de parution la plus ancienne d'un album **que je possède ou non mais enregistré dans la base de données**.

Exercice 2.7 : Écrire une requête permettant de déterminer le nombre d'albums que je possède.

Exercice 2.8 : Écrire une requête permettant de déterminer l'identifiant de la série dont je possède le plus d'albums (en cas d'égalité une seule suffit).

Exercice 2.9 : Écrire une requête permettant de déterminer le nombre de collections et d'albums par série (le titre de la série devant être fourni avec).

Partie 3 : Base de données « auto-école »

On considère la base de données suivante, utilisée par une auto-école afin de gérer les sessions d'exercices au code. Les tables sont les suivantes, avec des attributs intuitifs (et l'identifiant de séance dans l'ordre chronologique) :

- `CLIENTS`, avec les attributs `Id` (entier, clé primaire), `NomPrenom` (chaîne de caractères) et d'autres informations qui ne nous intéressent pas ici ;
- `SEANCES`, avec les attributs `Id_seance` (entier, formant avec l'identifiant du client la clé primaire), `Date` (format spécifique ordonné), `Id_client` (entier, clé extérieure référant au client), `NbFautes` (entier) ;
- `REPONSES`, avec les attributs `Id_seance` (entier, clé extérieure référant à la séance), `Id_client` (entier, clé extérieure référant au client), `Question` (entier), `Reponses` (chaîne de caractères) ;
- `SOLUTIONS`, avec les attributs `Id_seance` (entier, clé extérieure référant à la séance), `Question` (entier), `Reponses` (chaîne de caractères) ;

Exercice 3.1 : Quelle clé primaire peut-on envisager pour les tables `REPONSES` et `SOLUTIONS` ?

Exercice 3.2 : Il existerait un moyen de se passer de la table `SOLUTIONS`. Comment faire ? Par ailleurs, quel attribut est redondant dans le schéma ci-dessus ?

Exercice 3.3 : Écrire des requêtes permettant de récupérer les informations suivantes :

- Le nombre de clients enregistrés.
- L'ensemble des dates où une séance a eu lieu.
- Le nombre moyen de fautes d'un client précisé (on considère que son identifiant est disponible dans la valeur `n`).
- Le nombre minimal de fautes sur l'ensemble des clients et des séances.
- Le nombre maximal de questions ayant la même solution lors d'une séance précisée (on considère que son identifiant est disponible dans la valeur `s`).
- Le nombre de fautes qu'un client précisé (on considère que son identifiant est disponible dans la valeur `n`) a faites lors d'une séance précisée (de même, l'identifiant sera `s`). Pour cette dernière requête, on s'interdira d'utiliser la table `SEANCES` (l'idée est de faire une requête d'insertion à partir de données élémentaires).

Exercice 3.4 : Un client est considéré comme prêt à passer l'examen du code s'il a toujours fait moins de cinq fautes lors de ses trois précédentes séances. Écrire une requête qui détermine si c'est le cas d'un client précisé.

Exercice 3.5 : L'auto-école a décidé d'une offre de lancement promotionnelle pour fêter les deux mois de son activité : au client qui a fait le moins de fautes au total et ayant assisté à toutes les séances, dix heures de conduite sont offertes. Écrire une requête qui a permis de le trouver. On pourra supposer qu'il y a existence et unicité.