Exercices sur les BdD

Exercice 1 : L'énoncé de cet exercice utilise les mots du langage SQL suivants :

`SELECT FROM`, `WHERE`, `JOIN ON`, `INSERT INTO VALUES`, `UPDATE`, `SET`, `DELETE`, `COUNT`, `AND`, `OR`.

Pour la gestion des réservations clients, on dispose d'une base de données nommée « `gare` » dont le schéma relationnel est le suivant :

`Train` (`numT`, `provenance`, `destination`, `horraireArrivee`, `horraireDepart`)
`Reservation` (`numR`, `nomClient`, `prenomClient`, `prix`, `#numT`)

Les attributs soulignés sont des clés primaires. L'attribut précédé de # est une clé étrangère. La clé étrangère `Reservation.numT` fait référence à la clé primaire `Train.numT`.
Les attributs `horaireDepart` et `horaireArrivee` sont de type `TIME` et s'écrivent selon le format `"hh:mm"`, où `"hh"` représente les heures et `"mm"` les minutes.

1) Quel nom générique donne-t-on aux logiciels qui assurent, entre autres, la persistance des données, l'efficacité de traitement des requêtes et la sécurisation des accès pour les bases de données ?
2) a) On considère les requêtes SQL suivantes :

```
DELETE FROM Train WHERE numT = 1241;
DELETE FROM Reservation WHERE numT = 1241;
```

   Sachant que le train n°1241 a été enregistré dans la table `Train` et que des réservations pour ce train ont été enregistrées dans la table `Reservation`, expliquer pourquoi cette suite d'instructions renvoie une erreur.

   b) Citer un cas pour lequel l'insertion d'un enregistrement dans la table `Reservation` n'est pas possible.
3) Écrire des requêtes SQL correspondant à chacune des instructions suivantes :
   a) Donner tous les numéros des trains dont la destination est « Lyon ».
   b) Ajouter une réservation n°1307 de 33 € pour M. Alan Turing dans le train n°654.
   c) Suite à un changement, l'horaire d'arrivée du train n°7869 est programmé à 08h11. Mettre à jour la base de données en conséquence.
4) On rappelle qu'en SQL, la fonction d'agrégation `COUNT()` permet de compter le nombre d'enregistrements dans une table.
   Que permet de déterminer la requête suivante ?

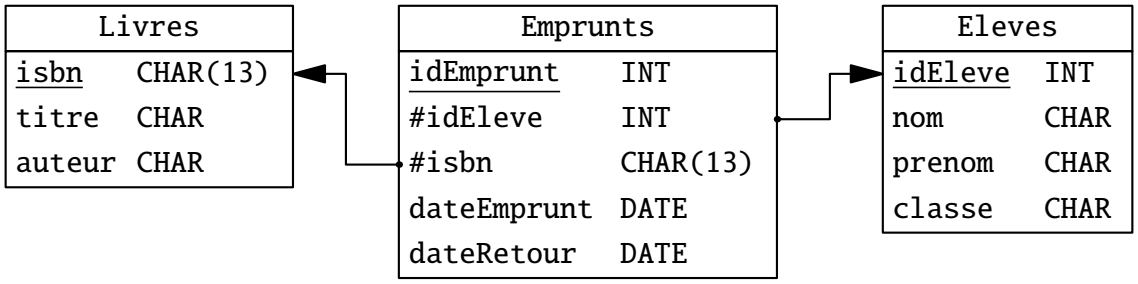
```
SELECT COUNT(*) FROM Reservation
WHERE nomClient = "Hopper" AND prenomClient = "Grace";
```

5) Écrire la requête qui renvoie les destinations et les prix des réservations effectuées par Grace Hopper.

Exercice 2 : L'énoncé de cet exercice utilise les mots du langage SQL suivants :

SELECT FROM, **WHERE**, **JOIN ON**, **INSERT INTO VALUES**, **UPDATE**, **SET**, **DELETE**, **COUNT**, **AND**, **OR**.

On considère dans cet exercice une gestion simplifiée des emprunts des ouvrages d'un CDI. La base de données utilisée sera constituée de trois relations (ou tables) nommées `Eleves`, `Livres` et `Emprunts` selon le schéma relationnel suivant :



Dans ce schéma relationnel, un attribut souligné indique qu'il s'agit d'une clé primaire. Le symbole # devant un attribut indique qu'il s'agit d'une clé étrangère et la flèche associée indique l'attribut référencé. Ainsi, l'attribut `idEleve` de la relation `Emprunts` est une clé étrangère qui fait référence à la clé primaire `idEleve` de la relation `Eleves`.

1) Expliquer pourquoi le code SQL ci-dessous provoque une erreur.

```
INSERT INTO Eleves VALUES (128, 'Dupont', 'Jean', 'T1');
INSERT INTO Eleves VALUES (200, 'Dupont', 'Jean', 'T1');
INSERT INTO Eleves VALUES (128, 'Dubois', 'Jean', 'T2');
```

2) Dans la définition de la relation `Emprunts`, qu'est-ce qui assure qu'on ne peut pas enregistrer un emprunt pour un élève qui n'a pas encore été inscrit dans la relation `Eleves` ?
3) Écrire une requête SQL qui renvoie les titres des ouvrages de Molière détenus par le CDI.
4) Décrire le résultat renvoyé par la requête ci-dessous.

```
SELECT COUNT(*)
FROM Eleves
WHERE classe = 'T2';
```

5) Camille a emprunté le livre Les misérables. Le code ci-dessous a permis d'enregistrer cet emprunt.

```
INSERT INTO Emprunts
VALUES (640, 192, '9782070409228', '2020-09-15', NULL);
```

Camille a restitué le livre le 30 septembre 2020.
Recopier et compléter la requête ci-dessous de manière à mettre à jour la date de retour dans la base de données.

```
UPDATE Emprunts SET . . . . . WHERE . . . . . ;
```

6) Décrire le résultat renvoyé par la requête ci-dessous.

```
SELECT DISTINCT nom, prenom
FROM Eleves, Emprunts
WHERE Eleves.idEleve=Emprunts.idEleve
AND Eleves.classe = 'T2';
```

```
SELECT DISTINCT nom, prenom
FROM Eleves JOIN Emprunts
ON Eleves.idEleve=Emprunts.idEleve
WHERE Eleves.classe = 'T2';
```

7) Écrire une requête SQL qui permet de lister les noms et prénoms des élèves qui ont emprunté le livre *Les misérables*.

Exercice 3 : Une ville souhaite gérer son parc de vélos en location partagée. L'ensemble de la flotte de vélos est stocké dans une table de données représentée en langage Python par un dictionnaire contenant des associations de type `id_velo : dict_velo` où `id_velo` est un nombre entier compris entre 1 et 199 qui correspond à l'identifiant unique du vélo et `dict_velo` est un dictionnaire dont les clés sont : `"type"`, `"etat"`, `"station"`.
Les valeurs associées aux clés `"type"`, `"etat"`, `"station"` de `dict_velo` sont de type *chaînes de caractères* ou *nombre entier* :

- `"type"` : chaîne de caractères qui peut prendre la valeur `"electrique"` ou `"classique"`
- `"état"` : nombre entier qui peut prendre la valeur 1 si le vélo est disponible, 0 si le vélo est en déplacement, -1 si le vélo est en panne
- `"station"` : chaînes de caractères qui identifie la station où est garé le vélo.

Dans le cas où le vélo est en déplacement ou en panne, "station" correspond à celle où il a été dernièrement stationné.
Voici un extrait de la table de données :

```
flotte = {
  12 : {"type" : "electrique", "etat" : 1, "station" : "Prefecture"},
  80 : {"type" : "classique", "etat" : 0, "station" : "Saint-Leu"},
  45 : {"type" : "classique", "etat" : 1, "station" : "Baraban"},
  41 : {"type" : "classique", "etat" : -1, "station" : "Citadelle"},
  26 : {"type" : "classique", "etat" : 1, "station" : "Coliseum"},
  28 : {"type" : "electrique", "etat" : 0, "station" : "Coliseum"},
  74 : {"type" : "electrique", "etat" : 1, "station" : "Jacobins"},
  13 : {"type" : "classique", "etat" : 0, "station" : "Citadelle"},
  83 : {"type" : "classique", "etat" : -1, "station" : "Saint-Leu"},
  22 : {"type" : "electrique", "etat" : -1, "station" : "Joffre"}
}
```

`flotte` étant une variable globale du programme.
Toutes les questions de cet exercice se réfèrent à l'extrait de la table flotte fourni ci-dessus.
Il y a des rappels sur les dictionnaires en langage Python à la fin de l'exercice.

1) a) Que renvoie l'instruction `flotte[26]` ?
   b) Que renvoie l'instruction `flotte[80]["etat"]` ?
   c) Que renvoie l'instruction `flotte[99]["etat"]` ?

2) Voici le script d'une fonction :

```
def proposition(choix):
  for v in flotte:
    if flotte[v]["type"] == choix and flotte[v]["etat"] == 1:
        return flotte[v]["station"]
```

   a) Quelles sont les valeurs possibles de la variable `choix` ?
   b) Expliquer ce que renvoie la fonction lorsque l'on choisit comme paramètre l'une des valeurs possibles de la variable `choix`.

3) a) Écrire un script en langage Python qui affiche les identifiants (`id_velo`) de tous les vélos disponibles à la station `"Citadelle"`.
   b) Écrire un script en langage Python qui permet d'afficher l'identifiant (`id_velo`) et la station de tous les vélos électriques qui ne sont pas en panne.

4) On dispose d'une table de données des positions GPS de toutes les stations, dont un extrait est donné ci-dessous. Cette table est stockée sous forme d'un dictionnaire.
Chaque élément du dictionnaire est du type :
`'nom de la station' : (latitude, longitude)`

```
stations = {
  'Prefecture' : (49.8905, 2.2967),
  'Saint-Leu' : (49.8982, 2.3017),
  'Coliseum' : (49.8942, 2.2874),
  'Jacobins' : (49.8912, 2.3016)
}
```

On admet que l'on dispose d'une fonction `distance(p1, p2)` permettant de renvoyer la distance en mètres entre deux positions données par leurs coordonnées GPS (latitude et longitude).
Cette fonction prend en paramètre deux tuples représentant les coordonnées des deux positions GPS et renvoie un nombre entier représentant cette distance en mètres.
Par exemple, `distance((49.8905, 2.2967), (49.8912, 2.3016))` renvoie `9591`.
Écrire une fonction qui prend en paramètre les coordonnées GPS de l'utilisateur sous forme d'un tuple et qui renvoie, pour chaque station située à moins de 800 mètres de l'utilisateur :

- le nom de la station ;
- la distance entre l'utilisateur et la station ;
- les identifiants des vélos disponibles dans cette station.

Une station où aucun vélo n'est disponible ne doit pas être affichée.

Rappels sur les dictionnaires en Python :

Action	Instruction et syntaxe
Créer un dictionnaire vide	`dico={}`
Obtenir un élément d'un dictionnaire existant à partir de sa clé et renvoie une erreur si cle n'existe pas dans le dictionnaire	`dico[cle]`
Modifier la valeur d'un élément d'un dictionnaire à partir de sa clé	`dico[cle]=nouvelle_valeur`
Ajouter un élément dans un dictionnaire existant	`dico[nouvelle_cle]=valeur`
Supprimer et obtenir un élément d'un dictionnaire à partir de sa clé	`dico.pop(cle)`
Tester l'appartenance d'un élément à un dictionnaire (renvoie un booléen)	`cle in dico`
Objet itérable contenant les clés (et qui n'est pas un objet de type `list`)	`dico.keys()`
Objet itérable contenant les valeurs (et qui n'est pas un objet de type `list`)	`dico.values()`
Objet itérable contenant les couples `(cle, valeur)`	`dico.items()`
Afficher les associations `cle:valeur` du dictionnaire `dico`	`for cle in dico:` `  print(cle, dico[cle])`