Corrigé du DS2

Exercice 1

Afin de lancer un nouveau service de streaming de musique, vous devez construire une base de données pour les morceaux de votre catalogue. Pour l'instant vous disposez d'une seule table avec les informations des morceaux. Voici **un extrait** de cette table :

Titre	Durée	Artiste	Album	Piste	CD	Année
Astronomy	384	Blue Öyster Cult	Secret Treaties	8	1	1974
Stone Cold Crazy	136	Queen	Sheer Heart attack	8	1	1974
Under Pressure	242	Queen and David Bowie	Hot Space	11	2	1982
The Outlaw Torn	589	Metallica	Load	14	1	1996
Fuel	270	Metallica	Reload	1	1	1997
The Memory Remains	279	Metallica and Marianne Faithfull	Reload	2	1	1997
Astronomy	398	Metallica	Garage Inc.	8	1	1998
Stone Cold Crazy	139	Metallica	Garage Inc.	11	2	1998
Fuel	276	Metallica and the San Francisco Symphony	S&M	6	1	1999
The Outlaw Torn	599	Metallica and the San Francisco Symphony	S&M	6	2	1999

Cette table ne convient pas vraiment pour faire une base de données.

1. Expliquer pourquoi aucune des colonnes ne peut pas servir de clef primaire.

Réponse

Dans chaque colonne, une valeur peut apparaître plusieurs fois : il peut y avoir plusieurs morceaux avec le même titre, la même durée, le même groupe *et cætera*.

2. Pourquoi est-ce que cette table est problématique si on veut rajouter des informations sur les artistes, comme leur nationalité ?

Réponse

Il peut y avoir plusieurs artistes, de nationalités différentes au sein d'un même morceau.

3. Quel est le problème si on souhaite chercher les morceaux d'un artiste ? Vous pourrez prendre l'exemple de Metallica.

Réponse

De même que précédemment, un artiste peut être associé à d'autres artistes. On pourrait imaginer une recherche avec un `WHERE ... LIKE '%Metallica%'` mais on risque de sélectionner d'autres artistes (on pourrait imaginer un groupe dénommé « We Are Not Metallica »).

Un ami vous suggère d'utiliser le schéma suivant :

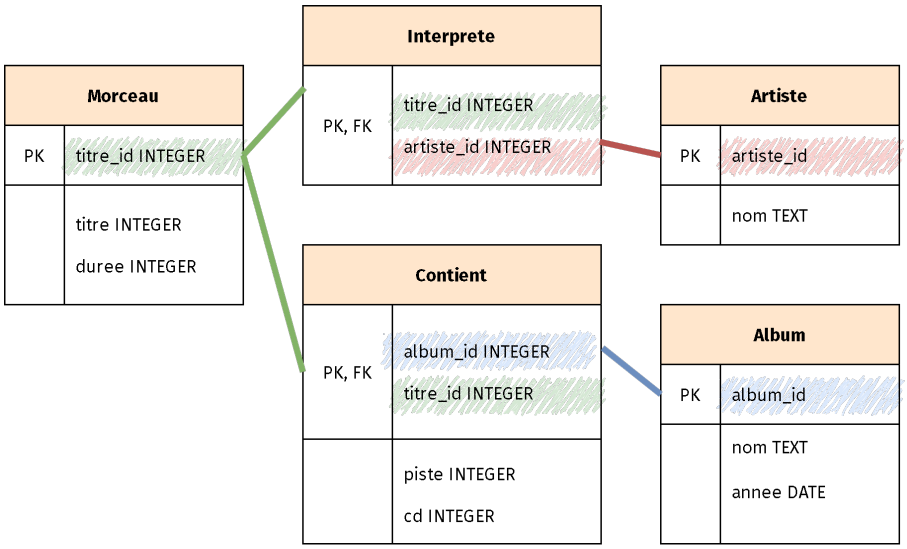
Morceau(titre_id, titre, duree, artiste_id, album, piste, cd, annee)
Artiste(artiste_id, nom)

4. Expliquer pourquoi cette représentation ne permet toujours pas de gérer les morceaux fait par deux artistes différents.

Réponse

Un titre est identifié de manière unique par `titre_id` et associé à un unique `artiste_id` : impossible de rajouter un autre artiste en gardant la même clé primaire.

Finalement, vous arrivez au schéma suivant :



5. Compléter les tables à l'aide des informations déjà disponibles. Un des morceaux n'a pas été intégré, inutile de l'y remettre. Si les noms dépassent, mettre uniquement le début.

titre_id	titre	duree
519	Astronomy	384
1219	Astronomy	398
316	Stone Cold Crazy	136
1319	Stone Cold Crazy	139
1298	Fuel	270
1570	Fuel	
401	Under Pressure	
1125	The Outlaw Torn	589
599	The Outlaw Torn	599

titre_id	artiste_id
519	25
1219	154
1319	154
1298	154
1570	154
1570	318
1125	154
1591	154
1591	318
316	79
401	79
401	108

artiste_id	nom
154	Metallica
318	San Francisco S.
25	Blue Öyster Cult
79	Queen
108	David Bowie

6. Comment appelle-t-on les clefs primaires de certaines tables apparaissant dans certaines tables, comme dans Interprete ?

Réponse

Ce sont des clés étrangères car elles font référence à des valeurs d'attributs d'autres tables.

7. Expliquer pourquoi le couple (titre_id, artiste_id) peut servir de clef primaire à Interprete.

Réponse

titre_id et artiste_id sont deux clés primaires respectivement de **Morceau** et **Artiste**, ainsi le couple (titre_id, artiste_id) identifie de manière unique un artiste donné interprétant un morceau donné.

8. Traduire en langage naturel les requêtes suivantes :

Code SQL

```sql
SELECT titre, duree FROM Morceau
WHERE duree > 600 ORDER BY duree DESC;
```

Réponse

Cette requête produit la table des titres et durées des morceaux d'une durée supérieure à 600 secondes, du plus long au plus court.

Code SQL

```sql
SELECT cd, piste, titre FROM Morceau
JOIN Contient ON Contient.titre_id = Morceau.titre_id
JOIN Album ON Contient.album_id = Album.album_id WHERE nom = "Garage Inc."
ORDER BY cd, piste;
```

Réponse

Cette requête produit la table du numéro de cd, numéro de piste et titres des morceaux de l'album « Garage Inc. », dans l'ordre des cd, et des numéros de pistes.

9. Donner la requête SQL permettant d'obtenir le nom de l'artiste dont l'identifiant est 200.

Code SQL

```sql
SELECT nom FROM Artiste
WHERE artiste_id = 200;
```

10. Donner la requête SQL permettant d'obtenir le nom de tous les albums sortis entre 1999 et 2010.

Code SQL

```sql
SELECT nom FROM Album
WHERE date BETWEEN 1999 AND 2010;
```

11. Donner la requête SQL permettant d'obtenir le titre et la durée de tous les morceaux, triés par ordre décroissant de durée, de tous les morceaux de l'artiste dont l'identifiant est 200.

Code SQL

```sql
SELECT titre, duree FROM Morceau
JOIN Interprete ON Interprete.titre_id = Morceau.titre_id
WHERE artiste_id = 200 ORDER BY duree DESC;
```

12. Les stars étant capricieuses, certaines veulent changer de nom. Donner la requête permettant à 'Maître Gims' de devenir 'Gims' dans la table des artistes.

Code SQL

```sql
UPDATE Artiste
SET name = 'Gims'
WHERE name = 'Maître Gims';
```

On rajoute maintenant les tables pour les utilisateurs :

Utilisateur(util_id INTEGER , nom TEXT, e-mail TEXT, adresse TEXT)
Ecoute(id_ecoute INTEGER, titre_id TEXT, util_id INTEGER, date DATE)

13. Expliquer pourquoi le couple (titre_id ,util_id) ne peut pas être une clef primaire.

Réponse

Un utilisateur donné peut écouter un titre donné à plusieurs dates différentes, ainsi ce couple peut apparaître dans plusieurs tuples différents.

14. Donner la requête SQL permettant d'ajouter l'utilisateur numéro 2179, qui s'appelle Bob VHS, dont l'email est bob.vhs@hotmail.com et qui habite à New York.

Code SQL

```sql
INSERT INTO Utilisateur VALUES
(2179, 'Bob VHS', 'bob.vhs@hotmail.com', 'New York')
```

15. Traduire le requête suivante en langage naturel :

Code SQL

```sql
SELECT COUNT(DISTINCT titre) FROM Morceau
JOIN Ecoute ON Morceau.titre_id = Ecoute.titre_id
WHERE date = "2020-12-12";
```

Réponse

Cette requête affiche le nombre de titres (distincts) écoutés le 12 décembre 2020.

Pour les exercices comportant des files et des piles, on utilisera les interfaces suivantes, qu'on suppose programmées en PYTHON :

Structure de file

- *file_vide()* renvoie une file vide ;
- *enfiler(file, valeur)* enfile la valeur en fin de file ;
- *defiler(file)* enlève la valeur en début de file et la renvoie ;
- *est_vide(file)* indique si la file est vide on non en renvoyant un booléen.

Structure de pile

- *pile_vide()* renvoie une pile vide ;
- *empiler(pile, valeur)* empile la valeur sur la pile ;
- *depiler(pile)* renvoie la valeur sur la pile et l'enlève de la pile ;
- *est_vide(pile)* indique si la pile est vide on non en renvoyant un booléen.

Exercice 2

Écrire en PYTHON une fonction `renverse_file` qui

- en entrée prend une file ;
- ne renvoie rien mais **utilise une pile** pour renverser la file.

Exemple d'utilisation

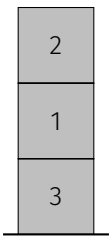
```
>>> print(F)
2 -> 3 -> 5 -> 1
>>> renverse_file(F)
>>> print(F)
1 -> 5 -> 3 -> 2
```

Réponse

```
def renverse_file(file):
    p = pile_vide()
    while not est_vide(file):
        empiler(p, defiler(file))
    while not est_vide(p):
        enfiler(file, depiler(p))
```

Exercice 3

Pour désigner une pile on donnera ses éléments en partant du sommet vers le fond. Ainsi `2 : 1 : 3` représentera la pile



On considère la fonction suivante :

Code Python

```
def mystere(pile1, pile2):
    if est_vide(pile1):
        return pile2
    else:
        empiler(pile2, depiler(pile1))
        return mystere(pile1, pile2)
```

1. Dans cette question on a `p = 30 : 20 : 10` et `q = 40 : 50 : 60`. Que renvoie l' appel `mystere(p,q)` ?

Réponse

Il renvoie `10 : 20 : 30 : 40 : 50 : 60`.

2. Expliquer en une phrase la fonction `mystere`.

Réponse

Cette fonction dépile la première pile pour l'empiler sur la deuxième.

3. Dans cette question on a `p = 10 : 20`.
Que se passe t-il lors de l'appel mystere(p , p) ?

Réponse

À ce moment la fonction dépile et empile à nouveau indéfiniment la valeur 10 sur `p`, jusqu'à saturer la pile de récursion.

Exercice 4

Écrire en PYTHON une fonction `max_file` qui

- en entrée prend une file **non vide** composée d'`int` **positifs** ;
- renvoie le maximum de cette file. Attention la file doit être remise dans l'état initial et aucune autre structure de données (pile, file, liste) ne doit être utilisée.

Exemple d'utilisation

```
>>> print(F)
>>> 2 -> 3 -> 5 -> 1
>>> max_file(F)
>>> 5
>>> print(F)
>>> 2 -> 3 -> 5 -> 1
```

Réponse

```
def max_queue(file):
    resultat = -1 # notre maximum temporaire
```

```
    enfiler(file, -1) # on enfile -1 pour savoir quand s'arrêter
    valeur = defiler(file)
    while valeur != -1:
        enfiler(file, valeur)
        if valeur > resultat:
            resultat = valeur
        valeur = defiler(file)
    return resultat
```

Voici ce qui se passe lors de l'utilisation du script :

```
from Alarm import Alarme

alarme1 = Alarme("Loritz", "971971971", False) # aucun événement
alarme2 = Alarme("Poincaré", "971971971", True) # idem
alarme1.intrusion() # date est interrogé et renvoie 0000 mais rien n'est consigné
alarme1.activer() # '0001 Activation ' est enregistré dans le journal de alarme1
alarme1.intrusion() # '0002 Intrusion envoi sms au 971971971' aussi
# '971971971 Loritz : 0002 Intrusion' est envoyé par SMS par alarme1
alarme1.desactiver() # '0003 Désactivation ' est enregistré dans le journal de alarme1
alarme2.intrusion() # '0004 Intrusion envoi sms au 971971971' est enregistré dans le
    journal de alarme2
# '971971971 Poincaré : 0004 Intrusion'  est envoyé par SMS par alarme1
```

Donc les SMS envoyés sont

- `'971971971 Loritz : 0002 Intrusion'`
- `'971971971 Poincaré : 0004 Intrusion'`

Et en définitive, `alarme1.journal` vaut :

```
['0001 Activation ', '0002 Intrusion envoi sms au 971971971', '0003 Désactivation ']
```

Dans le script donné en énoncé, on constate que la méthode `intrusion` est mal codée : on n'enregistre les évènements que lorsque l'alarme est active. Pour remédier à cela, il suffit de désindenter la ligne 20 : ainsi on consigne l'événement dans le journal quoi qu'il advienne.

Pour en plus gérer les envois de SMS échoués on aboutit au code suivant pour la méthode `intrusion` :

Code Python

```
def intrusion(self):
    evenement = date() + " Intrusion"
    if self.active:
        sms = self.lieu + ' : ' + evenement
        if envoie_sms(self.telephone, sms):
            evenement = evenement + " envoi sms au " + self.telephone
        else:
            evenement += " envoi au " + self.telephone + "échoué"
    self.journal.append(evenement)
```

Et pour vider le journal voici la méthode la plus simple :

Code Python

```python
def efface_journal(self):
    self.journal = []
```