Asie– 2024 – sujet2 - Correction

Exercice 3 (8 points)

Partie A

1. L'attribut `titre` ne peut pas être une clé primaire.

En effet, le titre *Muscle Museum* apparaît deux fois et le titre *Showbiz* apparaît également deux fois.

👉 *Commentaire : une clé primaire doit être unique pour chaque enregistrement.*

---

2. Résultat de la requête

```
SELECT titre, album
FROM Chanson
WHERE groupe = 'Guns N'Roses';
```

Résultat :

titre	album
Welcome too the Jungle	Appetite for Destruction

👉 *Commentaire : une seule chanson correspond au groupe indiqué.*

---

3. Requête pour obtenir les titres de l'album Showbiz dans l'ordre croissant

```
SELECT titre
FROM Chanson
WHERE album = 'Showbiz'
ORDER BY titre;
```

👉 *Commentaire : ORDER BY permet un tri lexicographique croissant par défaut.*

---

4. Requête pour ajouter la chanson Megalomania

```
INSERT INTO Chanson (titre, album, groupe)
VALUES ('Megalomania', 'Hullabaloo', 'Muse');
```

👉 *Commentaire : on précise les attributs si l'identifiant est auto-généré.*

---

5. Requête pour corriger la faute de frappe

```
UPDATE Chanson
SET titre = 'Welcome to the Jungle'
WHERE titre = 'Welcome too the Jungle';
```

👉 *Commentaire : UPDATE modifie les lignes correspondant à la condition.*

Partie B

6. Intérêt d'utiliser trois tables

Cela permet :

- d'éviter les redondances
- d'éviter les incohérences
- de faciliter les mises à jour
- de respecter le principe de normalisation

👉 *Commentaire : séparer les entités (Chanson, Album, Groupe) améliore l'intégrité des données.*

---

7. Rôle de l'attribut `id_album`

`id_album` est une clé étrangère qui fait référence à la clé primaire de la table Album.

👉 *Commentaire : il permet de relier chaque chanson à son album.*

---

8. Schéma relationnel

Chanson (**id** , titre, #id_album)
Album (**id** , titre, année, #id_groupe)
Groupe (**id** , nom)

👉 *Commentaire : les clés primaires sont soulignées et les clés étrangères précédées de #.*

---

9. Requête pour obtenir les albums contenant la chanson Showbiz

```
SELECT DISTINCT a.titre
FROM Album AS a
JOIN Chanson AS c ON c.id_album = a.id
WHERE c.titre = 'Showbiz';
```

👉 *Commentaire : DISTINCT évite les doublons.*

---

10. Requête pour obtenir les titres avec le nom de l'album des chansons du groupe Muse

```
SELECT c.titre, a.titre
FROM Chanson AS c
JOIN Album AS a ON c.id_album = a.id
JOIN Groupe AS g ON a.id_groupe = g.id
WHERE g.nom = 'Muse';
```

👉 *Commentaire : deux jointures sont nécessaires pour relier les trois tables.*

---

11. Description de la requête

```
SELECT COUNT(*) AS tot
FROM Album AS a
JOIN Groupe AS g ON a.id_groupe = g.id
WHERE g.nom = 'Muse';
```

Cette requête compte le nombre d'albums du groupe Muse.

👉 *Commentaire : COUNT()* renvoie le nombre de lignes correspondant à la condition.*

Partie C

12. Compléter les assertions

```
1 assert ordre_lex("", "a") == True
2 assert ordre_lex("b", "a") == False
3 assert ordre_lex("aaa", "aaba") == True
```

👉 *Commentaire : "aaa" précède "aaba" car à la quatrième lettre on compare fin de chaîne avec "b".*

---

13. Version récursive

```
def ordre_lex(mot1, mot2):
    if mot1 == "" and mot2 == "":
        return False
    if mot1 == "":
        return True
    if mot2 == "":
        return False
    if mot1[0] < mot2[0]:
        return True
    if mot1[0] > mot2[0]:
        return False
    return ordre_lex(mot1[1:], mot2[1:])
```

👉 *Commentaire : on compare les premiers caractères puis on appelle récursivement sur les suffixes.*

---

14. Version itérative

```
def ordre_lex(mot1, mot2):
    i = 0
    while i < len(mot1) and i < len(mot2):
        if mot1[i] < mot2[i]:
            return True
        if mot1[i] > mot2[i]:
            return False
        i += 1

    return len(mot1) < len(mot2)
```

👉 *Commentaire : si tous les caractères comparés sont égaux, le mot le plus court est le plus petit.*
