Université de Carthage Institut Préparatoire aux Etudes d'ingénieur de Nabeul Département :mathématique	IPEIN المعهد التحضيري للدراسات الهندسية بنابل Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul	Année universitaire : 2024/2025 Filière : SM/SP/ST Niveau d'étude : 2ème année Semestre : 2 Nombre de pages : 4 Date : 20/02/2025 Durée :1h30

DS Informatique N° 2

Une base de données est un élément central dans la gestion des informations d'un réseau social. Elle permet de stocker, organiser et traiter les données des utilisateurs, de leurs publications et des interactions entre eux. Dans ce contexte, SQLite sera utilisé comme système de gestion de base de données, et Python servira d'outil pour interagir avec celle-ci.

Les schémas relationnels suivants définissent l'organisation des données au sein d'une base de données destinée à un réseau social.

1. **Utilisateurs** (id, nom_utilisateur, email, date_inscription)

Cette table enregistre les informations essentielles sur chaque utilisateur inscrit sur le réseau.

- id: entier, identifiant unique de chaque utilisateur.
- nom_utilisateur: texte, le nom d'utilisateur.
- email: texte, l'adresse email de l'utilisateur.
- date_inscription: date, la date d'inscription de l'utilisateur.

2**. Publications** (id, # id_utilisateur, contenu, date_publication)

Cette table stocke les publications créées par les utilisateurs.

- id: entier, identifiant unique de chaque publication.
- id_utilisateur: entier, identifiant de l'utilisateur qui a publié, clé étrangère référence la colonne id de la table utilisateurs.
- contenu: texte, le contenu de la publication.
- date_publication: date, la date de publication.

3. **Interactions** (id, # id_utilisateur, # id_publication, type_interaction, date_interaction)

Cette table recense les interactions des utilisateurs avec les publications.

- id: entier, identifiant unique de chaque interaction.
- id_utilisateur: entier, identifiant de l'utilisateur qui interagit, clé étrangère référence la colonne id de la table utilisateurs.
- id_publication: entier, identifiant de la publication, clé étrangère référence la colonne id de la table publications.

- type_interaction: texte, le type d'interaction (like, comment, share).
- date_interaction: date, la date de l'interaction.

4. **Commentaires** (id, # id_utilisateur, # id_publication, contenu, date_commentaire)

Cette table permet de stocker les commentaires laissés par les utilisateurs sur les publications.

- id: Identifiant unique du commentaire.
- id_utilisateur: Référence l'utilisateur qui a commenté.
- id_publication: Référence la publication commentée.
- contenu: Texte du commentaire.
- date_commentaire: Date du commentaire.

5. **Amis** (id, id_utilisateur_1, id_utilisateur_2, date_amitie)

Cette table gère les relations d'amitié entre les utilisateurs.

- id: Identifiant unique de la relation d'amitié.
- id_utilisateur_1: Premier utilisateur.
- id_utilisateur_2: Deuxième utilisateur.
- date_amitie: Date de l'ajout en ami.

Questions :

Partie 1 : Algèbre Relationnelle (2 points)

Répondre en algèbre relationnelle aux questions suivantes :

1. Listez uniquement les noms d'utilisateurs et les dates d'inscriptions de tous les utilisateurs.
2. Récupérez les commentaires avec le nom d'utilisateur de la personne qui les a postés.
3. Trouvez les utilisateurs qui n'ont jamais publié de contenu.
4. Décrivez le rôle de cette requête :

$$R1 = Interactions \bowtie_{Interactions.id_{utilisateur} = Utilisateurs.id} Utilisateur$$

$$R2 = \pi_{nom_{utilisateur}} \left(\sigma_{type_{interaction} ¿' like'} (R1) \right)$$

Partie 2 : SQL (13 points)

1. Écrivez une requête SQL permettant de créer la table **Utilisateurs** avec ses contraintes (clé primaire et types de données appropriés).
2. La table **Publications** ne contient pas d'indication sur la visibilité des publications (publique ou privée). Écrivez une requête SQL pour ajouter une colonne visibilité (de type TEXT) à la table **Publications**.

3. Écrivez une requête SQL permettant d'ajouter un nouvel utilisateur dans la table Utilisateurs avec les informations suivantes : id = 101, nom_utilisateur = 'JD123', email = 'jamel123@email.com', date_inscription = '2025-02-01'.
4. Supprimez toutes les interactions liées à la publication dont id = 55.
5. Calculer et afficher le nombre total de publications par utilisateur.
6. Trouvez les publications créées après le 1er janvier 2025 ET contenant le mot 'promotion' dans leur contenu.
7. Affichez la liste des publications avec le nom d'utilisateur de la personne qui les a publiées.
8. Trouvez le nombre total de publications effectuées par chaque utilisateur et affichez les résultats avec nom_utilisateur.
9. Affichez les noms des utilisateurs ayant publié au moins 5 publications.
10. Affichez les 5 dernières publications publiées sur le réseau social, triées de la plus récente à la plus ancienne.
11. Affichez toutes les informations de l'utilisateur qui a créé le plus de publications.
12. Affichez la liste des publications créées par les utilisateurs ayant publié au moins 3 fois.
13. Quel est le nombre total d'interactions (likes, commentaires, partages) pour chaque publication ?
14. Décrivez le rôle de cette requête :

```
SELECT nom_utilisateur
FROM Utilisateurs
WHERE id IN (
    SELECT id_utilisateur
    FROM Interactions
    WHERE type_interaction = 'like'
    EXCEPT
    SELECT id_utilisateur
    FROM Interactions
    WHERE type_interaction = 'comment'
);
```

Partie 3 : Manipulation de la Base avec Python (SQLite3) (5 points)

1. Écrivez une fonction **open(nom_base)** qui se connecte à une base de données SQLite **nom_base** et retourne un objet de type **cursor**.
2. Écrivez une fonction **recuperer_publications(id_utilisateur)** qui retourne toutes les publications d'un utilisateur donné.
3. Écrivez une fonction **supprimer_utilisateur(id_utilisateur)** qui supprime un utilisateur et toutes ses publications, toutes ses interactions et toutes ses amitiés.
   Indication : Respectez l'ordre des suppressions pour éviter les conflits de clés étrangères.

4. Écrivez une fonction **close()** qui enregistre les modifications établies et ferme la connexion avec la base.
5. Écrivez un script Python permettant de
    - Charger le module sqlite3 et établir une connexion avec la base « **reseau_social.db** »
    - Affiche toutes ses publications.
    - Enregistrer les modifications établies et fermer la connexion avec la base.

6. Selon vous, comment convertir des données d'un modèle relationnel en un modèle orienté objet ? Donnez un exemple pour illustrer votre réponse.