

Devoir de contrôle n°2

Matière : INFORMATIQUE

Classe : 2^{ème} Année BG Durée : 1h

Respecter les notations de l'énoncé

Le sujet comporte 3 parties : Algèbre relationnelle, SQL, Python et le module sqlite3

Nous nous intéressons à une base de données relationnelle permettant de gérer un réseau social.

Le schéma relationnel de la base de données décrit ci-dessous a été élaboré pour les besoins de l'énoncé :

- **Utilisateur(idUt, nom, prenom, email, dateInscri, pays)**

La table *Utilisateur* enregistre les informations relatives aux utilisateurs de notre réseau social.

- idUt : identifiant de l'utilisateur de type entier, clé primaire
- nom : nom de l'utilisateur de type chaîne de caractères
- prenom : prénom de l'utilisateur de type chaîne de caractères
- email : email de l'utilisateur de type chaîne de caractères
- dateInscri : date d'inscription de type date
- pays : pays de type chaîne de caractères

- **Publication(idPub, idUt #, dateC, titre, contenu)**

La table *Publication* enregistre les informations relatives aux publications faites sur notre réseau social.

- idPub : identifiant d'une publication de type entier, clé primaire
- idUt : identifiant de l'utilisateur qui a créé la publication, de type entier, clé étrangère qui fait référence à la table *Utilisateur*
- dateC : date de création de la publication de type date
- titre : titre de la publication de type chaîne de caractères
- contenu : contenu de type chaîne de caractères

- **Interaction(idInt, idUt #, idPub #, typeInt, dateInt)**

La table *Interaction* enregistre les informations relatives aux interactions des utilisateurs avec les publications.

- idInt : identifiant d'une interaction de type entier, clé primaire
- idUt : identifiant de l'utilisateur qui interagit, de type entier, clé étrangère qui fait référence à la table *Utilisateur*

- idPub : identifiant de la publication, de type entier, clé étrangère qui fait référence à la table *Publication*.
- typeInt : le type de l'interaction qui peut être 'Like', 'Aime', 'Partage' (de type chaîne de caractères)
- dateInt: date de l'interaction de type date
- **Amis(idUt1 #, idUt2 #, etat)**

La table *Amis* enregistre les informations relatives aux demandes d'ajout d'amis.

- idUt1 : identifiant de l'utilisateur qui demande l'amitié, de type entier, clé étrangère qui fait référence à la table *Utilisateur*
- idUt2 : identifiant de l'ami souhaité, de type entier, clé étrangère qui fait référence à la table *Utilisateur*
- idUt1 et idUt2 forment une clé primaire
- etat : état de la relation qui peut être en 'Attente' ou 'Confirmé', de type chaîne de caractères
- **Message(idExp #, idRec #, nbre)**

La table *Messages* permet de compter le nombre de messages envoyés entre utilisateurs.

- idExp : identifiant de l'utilisateur qui envoie le message, de type entier, clé étrangère qui fait référence à la table *Utilisateur*
- idRec : identifiant de l'utilisateur recevant le message, de type entier, clé étrangère qui fait référence à la table *Utilisateur*
- idExp et idRec forment une clé primaire
- nbre : nombre de messages envoyés de *idExp* vers *idRec*, de type entier supérieur à zéro

Règles de gestion et contraintes à considérer pour des raisons de simplicité

- Le type date est au format "AAAA-MM-JJ"
- Les utilisateurs sont identifiés par leurs numéros de carte d'identité considérés comme entier.
- Un utilisateur peut : avoir des amis, créer des publications, faire des interactions à des publications, envoyer message(s) à un/des utilisateur(s) du réseau.
- En cas de demande d'amitié, on insère une nouvelle ligne dans la table *Amis*. L'identifiant de l'utilisateur qui demande l'ajout d'amis est mis dans la colonne *idUt1*, l'identifiant de l'ami souhaité est mis en tant que *idUt2* et l'état est mis à 'Attente' en attendant la confirmation d'ajout. En cas de confirmation, on met à jour la table (état est mis à 'Confirmé') sans faire insertion de nouvelle ligne.
- En cas d'**envoi de message** d'un utilisateur (expéditeur) vers un autre (receveur), on examine la table *Message*. Si le couple identifiant de l'expéditeur et identifiant du receveur n'existe pas alors on ajoute une nouvelle ligne avec l'identifiant de l'expéditeur, l'identifiant du receveur et 1 (nbre=1). Sinon, on incrémente la valeur de nombre par 1.

Partie 1

Exprimer en algèbre relationnelle les requêtes permettant de :

- 1- Afficher les titres des publications créées par des Tunisiens le 10 octobre 2024.
- 2- Afficher les noms, prénoms et pays des utilisateurs sans amis. **Indication** : le champ état doit être à 'Confirmé' pour considérer le lien d'amitié.

Partie 2

Exprimer en SQL les requêtes suivantes permettant de :

- 1- Afficher les noms, prénoms et pays des utilisateurs ayant des publications en 2025 avec le mot 'Palestine' dans le titre.
- 2- Supprimer les publications créées avant 2000 et n'ayant pas d'interaction.
Indication pour 1 et 2 : On peut utiliser *dateC like '2025%'* pour vérifier l'année.
- 3- Afficher les identifiants, noms et pays des utilisateurs ayant plus que 10 publications par jour.
- 4- Afficher les identifiants des expéditeurs qui n'ont reçu aucun message de la part de leur receveur.
Indication : on peut utiliser une condition avec la syntaxe (*champs1,champs2*) *in (val1, val2)*
- 5- Détecter les 5 jours de pics de publications sur notre réseau social en 2025. On affiche date et nombre de publications.

Partie 3

Dans la suite, les fonctions demandées doivent être écrites en Python. On désigne par *cur* le curseur d'exécution de requêtes.

- 1- Ecrire la fonction **nbMessage** qui prend comme paramètres :
 - *cur* : le curseur d'exécution de requêtes
 - *idU1* : l'identifiant de l'utilisateur expéditeur
 - *idU2* : l'identifiant de l'utilisateur receveur (destination du message)

et retourne le nombre de messages envoyés de *idU1* vers *idU2* et -1 si pas de messages.

- 2- Ecrire la fonction **envoiMessage** qui prend comme paramètres :
 - *cur* : le curseur d'exécution de requêtes
 - *idU1* : l'identifiant de l'utilisateur expéditeur
 - *idU2* : l'identifiant de l'utilisateur receveur (destination du message)

et met à jour la table Messages selon les règles de gestion indiquées à la page 2.

Indication : Utiliser la fonction *nbMessage*.

- 3- Ecrire la fonction **nbInteraction** qui prend comme paramètres :
 - *cur* : le curseur d'exécution de requêtes
 - *typeInt* : le type d'interaction

et retourne le nombre d'interactions de type *typeInt* par publication. Le retour de cette fonction est une liste de tuples. Par exemple, si *typeInt*='Like', cette fonction retourne une liste de tuples où chaque tuple contient l'identifiant d'une publication ainsi que le nombre de 'Like' pour cette publication.

4- Ecrire la fonction **detailPub** qui prend comme paramètres :

- *cur* : le curseur d'exécution de requêtes
- *idPub* : l'identifiant d'une publication

et affiche la date et le titre de la publication correspondante à *idPub*.

5- Ecrire le script Python permettant de :

- Importer le module *sqlite3*
- Se connecter à la base de données 'ReseauSo.db'
- Créer le curseur *cur* d'exécution
- Chercher la date et le titre de la publication ayant le maximum de nombre de 'Like'.

Indication : Utiliser la fonction *nbInteraction(cur, 'Like')*, chercher le maximum puis utiliser la fonction *detailPub*.