— **Examen blanc :** préparation à l'examen de session 2
— Durée : 2 heures.
— Autorisé : une feuille A4 de notes personnelles approuvée par les surveillants.
— Tout autre document, ordinateur, calculatrice ou téléphone est interdit.

Partie 1 : Forum d'entraide automodéré

Pour toute cette partie, on fixe une base de données utilisée pour gérer les utilisateurs d'un forum d'entraide ainsi que leurs questions et réponses. Le forum est automodéré, c'est-à-dire que les utilisateurs peuvent donner une note aux réponses, afin de faire ressortir les réponses de qualité et de faire disparaître les réponses hors sujet ou dont le contenu est inapproprié.

La base de données respecte le schéma ci-dessous, où les clefs primaires sont soulignées et les clefs étrangères listées à la suite.

```
personne(pseudo, nom, prenom, email, mdp)
question(idQ, auteurQ, dateQ, theme, description, resolue)
reponse(idR, auteurR, dateR, idQ, message)
evaluation(pseudo, idR, note)
```

FK : `auteurQ`, `auteurR` et `pseudo` dans `question`, `reponse` et `evaluation` font référence à `pseudo` dans `personne` ;
FK : `idQ` dans `reponse` fait référence à `idQ` dans `question` ;
FK : `idR` dans `evaluation` fait référence à `idR` dans `reponse`.

La table `personne` contient simplement les utilisateurs inscrits du forum, avec leur pseudonyme, nom, prénom, adresse e-mail et mot de passe. La table `question` contient les questions posées par les utilisateurs. Pour chaque question, on connaît son auteur, sa date et son thème. L'attribut `description` est un texte qui décrit le problème rencontré par l'utilisateur. L'attribut `resolue` est un booléen qui indique si la question est encore ouverte (faux) ou a reçu une réponse satisfaisante (vrai). La table `reponse` contient les réponses apportées aux questions. Pour chaque réponse, on connaît son auteur, sa date et l'identifiant (`idQ`) de la question concernée par la réponse. L'attribut `message` contient le texte de la réponse. Finalement la table `evaluation` contient les notes (entre 0 et 5) attribuées aux réponses par les utilisateurs.

▸ Exercice 1 : Triggers

Dans chacun des cas suivants, dites si un trigger est nécessaire pour obtenir le comportement voulu. Si oui, donnez la commande de création du trigger et décrivez en français la procédure trigger correspondante, en précisant sa valeur de retour. Si non,

expliquez comment obtenir le comportement voulu, par exemple en faisant les bons choix à la création de la table.

1. Les descriptions des questions ne peuvent pas excéder 5000 caractères.
2. À l'insertion d'une nouvelle réponse, si la date n'est pas spécifiée, elle est automatiquement fixée à la date du jour.
3. Lorsqu'une réponse reçoit sa 10ème note inférieure à 1, elle est automatiquement supprimée.
4. Une question qui a été marquée comme résolue ne peut pas être ouverte à nouveau.
5. Un utilisateur ne peut pas poser une nouvelle question s'il est l'auteur d'une question qui est encore ouverte.
6. Un utilisateur ne peut pas évaluer deux fois la même réponse.
7. Lorsque la description d'une question est modifiée, le texte "Édité le (date)." est automatiquement ajouté à la fin du message, où (date) est remplacé par la date du jour.
8. Lorsqu'une ou plusieurs nouvelles questions sont ajoutées sur le forum, une notification est produite, indiquant : "De nouvelles questions ont été ajoutées le (date)."
9. Seules les questions ouvertes peuvent recevoir de nouvelles réponses.
10. Les utilisateurs ne peuvent pas modifier leur évaluation d'une réponse.

▸ Exercice 2 : Programmation de trigger

On souhaite implémenter un trigger permettant de marquer automatiquement comme résolues les questions ayant reçu une réponse évaluée par au moins 10 utilisateurs et dont la moyenne des notes est supérieure à 4. Dans ce cas, une notification sera également produite, indiquant que "La question `(idQ)` a été résolue par `(auteurR)` !"

Proposez une commande pour créer le trigger, et donnez le code de la procédure correspondante en PL/pgSQL.

▸ Exercice 3 : Transactions

Un administrateur souhaite effectuer la requête suivante :

```
BEGIN;
DELETE FROM question
WHERE NOT EXISTS (
  SELECT * FROM reponse NATURAL JOIN evaluation
  WHERE reponse.idQ = question.idQ AND note = 5
);
UPDATE question SET resolue = True
WHERE EXISTS (
  SELECT * FROM reponse NATURAL JOIN evaluation
  WHERE reponse.idQ = question.idQ AND note = 5
);
COMMIT;
```

En même temps, l'utilisateur 'HP.otter' souhaite effectuer la requête suivante :

```
DELETE FROM evaluation
WHERE pseudo = 'HP.otter';
```

1. Que peut-on dire de la valeur `resolue` des questions présentes dans la base de données après n'importe quelle exécution en série des deux requêtes ?
2. Donnez un scénario dans lequel une exécution en parallèle des deux requêtes conduit à une erreur de sérialisabilité. Justifiez votre réponse en expliquant et en nommant l'erreur mise en évidence.
3. Dans quel niveau d'isolation doit-on se placer pour que l'erreur mise en évidence ne puisse pas se produire ?

▸ **Exercice 4 : Ordres de grandeur**

On s'intéresse au stockage de la table `personne`.
Justifiez vos réponses en décomposant les étapes de vos estimations.

1. Donnez un ordre de grandeur de la taille d'un enregistrement de cette table.

On estime que le forum servira au maximum quelques milliers d'utilisateurs.

2. Donnez un ordre de grandeur de l'espace total occupé par la table.
3. Donnez un ordre de grandeur du nombre de blocs nécessaires pour stocker la table.
4. Donnez un ordre de grandeur du nombre d'entrées d'un index primaire non dense construit sur l'attribut 'pseudo' et de l'espace nécessaire pour stocker l'index.

Partie 2 : Questions de cours

▸ **Exercice 5 : Petites questions**

1. Justifiez que la complexité des opérations sur une base de données se mesure en nombre d'accès au disque.
2. Quelle est la différence principale entre un index primaire et un index secondaire ?
3. Donnez un scénario illustrant l'intérêt de l'*atomicité* des transactions.
4. Pour quelle raison n'utilise-t-on pas systématiquement le niveau d'isolation maximum pour les transactions ?

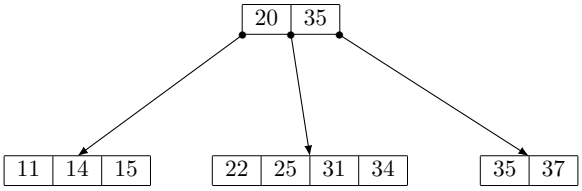
▸ **Exercice 6 : Sérialisabilité par conflits**

Les ordonnancements suivants sont-ils sérialisables par conflit ? Si oui, donnez une exécution en série équivalente. Si non, justifiez votre réponse.

1. $R_1(x)\ R_4(x)\ W_4(x)\ R_1(y)\ R_4(z)\ W_4(z)\ W_3(y)\ W_3(z)\ W_1(t)\ W_2(z)\ W_2(t)$
2. $R_1(x)\ R_1(t)\ R_3(z)\ R_4(z)\ W_2(z)\ R_4(x)\ R_3(x)\ W_4(x)\ W_4(y)\ W_3(y)\ W_1(y)\ W_2(t)$

▸ **Exercice 7 : Arbres B+**

On considère l'arbre B+ ci-dessous dont les clefs sont des entiers. On suppose qu'un bloc du disque peut contenir au maximum 4 entiers et 5 pointeurs.



Représentez l'évolution de l'arbre pour les opérations suivantes :
(Repartez de l'arbre de départ pour chaque question)

1. Insertion de 36.
2. Insertion de 26.
3. Suppression de 20.
4. Suppression de 35.