Projet : partie 2

Rappel : consignes générales

Voici la partie 2 de votre projet.

Comment ?
Le mot clef de la réalisation de votre projet est : l'autonomie. Il est recommandé en premier lieu de favoriser les recherches et le travail en groupe.

Qui ?
Toujours le même groupe que pour la partie 1 du projet.

Quand ?
Les différentes étapes du projet sont les suivantes :
- Présentation du projet : semaine du 6 mars
- Séance de suivi du projet : semaine du 27 mars
- Soutenance du projet : semaine du 10 avril

Quoi ?
Le projet doit être rendu sur Moodle le jour de votre soutenance et doit contenir notamment :
- Un rapport ;
- les scripts SQL qui vous ont permis de faire le projet ;
- Autres fichiers (explications diverses, MCD, MLD etc)

Concernant le rapport, il doit être structuré comme suit :
• Page de garde :
Le titre du projet, les noms des membres de l'équipe.
• Introduction :
Présentation succincte du projet.
• Contenu du rapport
Le contenu du rapport :
- *synthétiser votre travail et vos réalisations*
- *exposer les difficultés rencontrées et les solutions adoptées*
- *mettre en valeur votre travail et vous évaluer (points forts du projet, points à améliorer)*
- *expliquer la gestion du projet (répartition des tâches, outils utilisés pour travailler en groupe).*

• Conclusion
Bilan du projet, apports individuels et collectifs, etc.

Il est important de faire attention à l'orthographe.

Partie 1 - Influence d'un index

1. Expliquer à quoi sert un index en base de données.

2. Trouver les requêtes SQL afin de :

- compter le nombre d'étudiants qui sont dans la maison "Gryffindor" ;

- mesurer le temps de la requête avec la commande SHOW PROFILE (https://dev.mysql.com/doc/refman/8.0/en/show-profile.html) ;

- ajouter un index sur la colonne "house_id" de la table "students" ;

- mesurer à nouveau le temps de la requête après l'ajout de l'index ;

- mesurer à nouveau le temps de la requête mais sans index.

Attention : dans MySQL, un index est utilisé par défaut par le SGBD pour optimiser les requêtes. Si vous souhaitez mesurer les performances d'une requête sans utiliser l'index que vous venez de créer, vous pouvez utiliser `IGNORE INDEX`.

3. Pour les requêtes suivantes, vous devez dire à quoi correspond chaque requête. Ensuite, vous devez mesurer le temps de la requête, rajouter un index, mesurer encore une fois le temps de la requête. Si besoin, adapter ces requêtes à votre base de données.

Requête a

```
SELECT houses.house_name, courses.course_name, COUNT(*) AS
num_students
FROM students
JOIN houses ON students.house_id = houses.house_id
JOIN courses ON students.course_id = courses.course_id
GROUP BY houses.house_name, courses.course_name
ORDER BY num_students DESC;
```

Requête b

```
SELECT student_name, email
FROM students
WHERE course_id IS NULL;
```

Requête c

```
SELECT houses.house_name, COUNT(*) AS num_students
FROM students
JOIN houses ON students.house_id = houses.house_id
```

```
WHERE EXISTS (
  SELECT *
  FROM courses
  WHERE course_name IN ('Potions', 'Sortilèges', 'Botanique')
    AND course_id = students.course_id
)
GROUP BY houses.house_name;
```

Requête d

```
SELECT s.student_name, s.email
FROM students s
JOIN (
  SELECT student_id, year_id, COUNT(DISTINCT course_id) AS
num_courses
  FROM students
  GROUP BY student_id, year_id
) AS sub
ON s.student_id = sub.student_id AND s.year_id = sub.year_id
JOIN (
  SELECT year_id, COUNT(DISTINCT course_id) AS num_courses
  FROM students
  GROUP BY year_id
) AS total
ON s.year_id = total.year_id AND sub.num_courses =
total.num_courses
WHERE sub.num_courses = total.num_courses;
```

Partie 2 – Les vues

1. Expliquer à quoi sert une vue en base de données. Expliquer quelle est la différence entre une vue logique et une vue matérialisée.

2. Trouver les requêtes SQL afin de :

   a. Créer une vue logique qui affiche le nom, l'email et la maison de chaque étudiant qui suit un cours de potions.

   b. Afficher le résultat de la vue.

   c. Rajouter 2 étudiants qui suivent un cours de potion.

   d. Afficher (encore) le résultat de la vue.

3. Modification interdite d'une vue :

Créer une vue `house_student_count` qui regroupe les étudiants par maison et compte le nombre d'étudiants dans chaque maison.

Il y aura donc 2 colonnes : une colonne `house_name` avec le nom des maisons et une colonne `student_count` avec le nombre d'étudiants par maison.

Cette vue va utiliser des fonctions d'agrégation (`COUNT` et `GROUP BY`).

Essayer de modifier la colonne contenant le nombre d'étudiants dans une maison. Par exemple, pour la maison Gryffondor, définir le nombre d'étudiants à 10.

Est-ce que cette requête génère une erreur ? Et si la vue `house_student_count` avait été une table normale, est-ce que cette requête aurait fonctionné ?

Partie 3 – Procédure stockée et trigger

1. Expliquer à quoi sert les procédures stockées et les triggers.

2. Création d'une vue matérialisée pour `house_student_count` à l'aide d'une procédure stockée sur MySQL

Dans cet exercice, vous allez créer une vue matérialisée pour la vue logique `house_student_count` que vous avons utilisée précédemment. Pour créer la vue matérialisée, nous allons utiliser une procédure stockée sur MySQL.

En effet, les vues matérialisées ne sont pas prises en charge par défaut sur MySQL, contrairement à d'autres systèmes de gestion de base de données comme PostgreSQL ou Oracle.

Cependant, vous pouvez simuler une vue matérialisée en utilisant une table temporaire et une procédure stockée.

a. Créez une table `house_student_count_materialized` qui sera notre vue matérialisée

Cette table va donc avoir la même structure que la vue logique `house_student_count`.

Il y aura donc 2 colonnes : une colonne `house_name` avec le nom des maisons et une colonne `student_count` avec le nombre d'étudiants par maison.

b. Créez une procédure stockée pour rafraîchir la vue matérialisée `house_student_count_materialized`

La procédure stockée doit s'appeler `refresh_house_student_count_materialized`. Elle doit permettre de mettre à jour la table `house_student_count_materialized` en recalculant et en insérant le nombre d'étudiants pour chaque maison à partir des données présentes dans les tables `students` et `houses`.

La procédure stockée doit donc effectuer les actions suivantes :

- Elle vide la table `house_student_count_materialized` en utilisant la commande `TRUNCATE TABLE`. Cette étape supprime toutes les données existantes de la table.

- Ensuite, elle insère de nouvelles données dans la table `house_student_count_materialized`.

c. Exécutez la procédure stockée pour rafraîchir la vue matérialisée

Maintenant, la table `house_student_count_materialized` contient les mêmes informations que la vue `house_student_count`. Vous pouvez exécuter des requêtes sur cette table comme vous le feriez sur une vue matérialisée.

3. Mise à jour de la vue matérialisée `house_student_count_materialized`

   a. Ajoutez un nouvel étudiant à la table `students`

   b. Affichez le contenu de la table `house_student_count_materialized` pour vérifier si le nouvel étudiant a été pris en compte

   c. Exécutez la procédure stockée `refresh_house_student_count_materialized()` pour mettre à jour les données de la vue matérialisée `house_student_count_materialized`

   d. Affichez à nouveau le contenu de la vue matérialisée `house_student_count_materialized` pour vérifier si le nouvel étudiant a été pris en compte après l'exécution de la procédure stockée

4. Création de triggers sur la vue matérialisée `house_student_count_materialized`

Dans cet exercice, vous allez créer des triggers pour mettre à jour automatiquement la vue matérialisée `house_student_count_materialized` chaque fois qu'un étudiant est ajouté ou supprimé.

Cela vous permettra de garder la vue matérialisée à jour sans avoir à appeler manuellement la procédure stockée `refresh_house_student_count_materialized()`.

   a. Créez un trigger `AFTER INSERT` pour mettre à jour automatiquement la vue matérialisée `house_student_count_materialized` chaque fois qu'un étudiant est ajouté dans la base de données

   b. Créez un trigger `AFTER DELETE` pour mettre à jour automatiquement la vue matérialisée `house_student_count_materialized` chaque fois qu'un étudiant est supprimé dans la base de données

5. Tester les triggers de la vue matérialisée `house_student_count_materialized`

   a. Affichez le contenu de la table `house_student_count_materialized` avant d'effectuer des modifications

   b. Insérez un nouvel étudiant dans la table `students`

   c. Affichez le contenu de la table `house_student_count_materialized` après l'insertion pour vérifier si le trigger `AFTER INSERT` a fonctionné

   d. Supprimez l'étudiant précédemment inséré de la table `students`

   e. Affichez le contenu de la table `house_student_count_materialized` après la suppression pour vérifier si le trigger `AFTER DELETE` a fonctionné