D1PM

Mise à jour par id

Une fonctionnalité à été demandé par vos utilisateurs. Ils souhaiteraient pouvoir changer de nom.

Rédigez une requête SQL pour changer de nom en fonction de l'id.

Mise à jour multiple

Suite à l'ajout d'un nouveau rôle administrateur d'équipe (nommé admin). Nos utilisateurs admin (gestion du site) doivent passer au role "superadmin"

Mettez à jour tous les utilisateurs possédant le role "admin" en leur attribuant le nouveau role "superadmin"

> 📌 N'oubliez pas de rajouter "superadmin" a l'énumération de la structure

Mise à jour (avancé)

Suite à la mise en place d'adresse mail à domaine professionnel @monentreprise.fr pour nos superadmin. Il est souhaité que tous les utilisateurs "superadmin" puissent se connecter avec une adresse mail <name>@monentreprise.fr.

Utilisez la colonne name de la table user et mettez à jour les utilisateurs "superadmin" de la table pour correspondre à cette demande

Suppression d'un compte

Un utilisateur viens de demander la suppression de son compte.

son id est le n°3.

Une meilleure lecture

Les superadmin ont fait une demande: pouvoir lister les utilisateurs par ordre alphabétique via leur nom. Donnez leur un coup de main.

Une meilleure lecture 2

Ce serai bien que nos utilisateurs puissent profiter également d'un ordre d'affichage amélioré.

Affichez les tâches de la table todos par état de la tache, puis par date de création.

> 📌 Les tâches seront groupé par état les tâches possédant le même état seront trié par date de création.

L'équipe marketing entre en scène.

L'équipe marketing nous à demandé d'analyser notre base de donnée à la recherche de données facilitant leur travail:

"Le nombre total d'utilisateur (attention a ne pas compter les superadmin)"

"L'age moyen de nos utilisateurs"

"Trouver le plus jeune et le plus vieux de nos utilisateurs"

Une question de sécurité

Modifiez la table users pour rajouter une colonne `password` (pour le moment la colonne sera en clair, pas de chiffrement).

Il nous a été conseillé par un consultant de créer une view pour notre table utilisateur permettant de supprimer les informations sensible (tel que cette colonne password).

Créez une view `users_public_v` qui permettra d'afficher plus facilement le profil public de nos utilisateurs en supprimant les informations critique de sécurité.

> 📌 Une fois la vue créée, vous pouvez effectuer un SELECT sur notre vue comme une table normale.

Une question de sécurité 2

Oops, nos comptes superadmin sont affichés dans notre view,

Modifiez la vue `users_public_v` et ajoutez une clause WHERE pour enlever nos superadmins

Le retour de l'équipe marketing

L'équipe marketing souhaiterai accéder aux analytics précédemment demandé de manière plus régulière, sans passer par vous (histoire d'éviter de vous déranger), et de manière simple en une seule commande SQL:

```
SELECT users_count, users_age_average, users_older, users_younger FROM users_analytics_v;
```

Créez une vue récapitulant les données extraites de l'exercice "L'équipe marketing entre en scène." compatible avec la requête SQL précédente.

> 📌 N'oubliez pas vos alias AS

Ajout d'une liaison user↔todo

Il est temps de segmenter les taches de nos utilisateurs.

Effectuez un ALTER sur la table TODO afin d'ajouter une colonne `user_id` permettant de leur définir leur propriétaire. `user_id` sera notre clée étrangère (FOREIGN KEY)

A ce stade vous devriez avoir a minima la structure suivante (ne prend pas en compte les vues ni les contraintes):

```
users: {
  id: int primary auto_increment,
  name: var_char(255),
  email: var_char(255),
  role: enum("superadmin", "admin", "user"),
  age: int,
  birthday: Date,
  password: var_char(255)
}

todos: {
  id: int primary auto_increment,
  title: var_char(255),
  updated_at: Datetime,
  created_at: Datetime,
  state: ENUM("todo", "done"),
  user_id: int FOREIGN KEY REFERENCES users(id)
}
```

Si ce n'est pas le cas, effectuez vos ALTER en conséquence.

Créez votre dataset de todos, cela vous sera utile pour la suite.

Les todos de nos utilisateurs

La mise à jour de la fonctionnalité de gestion de tâche nécessite de récupérer toutes les tâches liées a un utilisateur (exemple l'utilisateur 2). Récupérez toutes les tâches de l'utilisateur 2.

📌 Les infos de l’utilisateur ne doivent pas apparaître dans les résultats

Les todos de nos utilisateurs 2

Une demande de fonctionnalité est arrivé sur votre bureau.

Créez une requête SQL qui vous retourne le nombre total de tâche non terminée pour un utilisateur donné.

Des todos perdues

Après avoir supprimé un utilisateur, il s’avère que nos entrées dans la table todos ne sont plus cohérentes avec la table users.

Il existe des todos qui sont reliés a des id utilisateurs qui ont été supprimées.

Paramétrez la contrainte d’intégrité sur la table todos pour supprimer les todos appartenant à un utilisateur quand celui ci est supprimé.