Etape 4: qualité globale de la base de données et du code source

Avantages liés au schéma de la base de données

La structure de la base de données est bien organisée. Elle contient des noms de tables spécifiques qui permettent une lecture claire et une compréhension de la relation entre les différentes tables.

1- La base de données utilise des clés primaires (malgré l'absence des clés étrangères) pour établir des liens entre les différentes tables. Cela permet de garantir la cohérence des données entre les tables et facilite également la gestion des requêtes complexes.

2- Les tables sont normalisées au maximum pour réduire au minimum la duplication de données et faciliter la maintenance et les mises à jour de la base données.

3- Les tables et les colonnes contiennent des noms qui facilitent la compréhension de leur contenu.

4- Des champs tels que "shared_user", "tags", et "search_hit" permettent de classer les données et de faciliter la recherche et la récupération de l'information désirée par les utilisateurs.

Ces éléments contribuent à une bonne organisation de la base de données qui est claire et facile à maintenir, même avec des volumes importants de données.

Inconvénients liés au schéma de la base de données

1- Il existe plusieurs colonnes calculées ceci pourrait démontrer un problème de normalisation de tables car à travers cela on pourrait avoir des données incohérentes et redondantes dans le système.

Les mises à jour des valeurs calculées peuvent devenir complexes, surtout si elles dépendent de plusieurs tables.

Exemple : dans la table stories, nous avons les colonnes telles que:

- comment_user_ids: Text(chaîne de caractères) reprenant l'id de tous les utilisateurs ayant commentés une storie,
- tags : format Text qui est une chaîne de caractères permettant de stocker la liste des tags associée à une storie,
- image_urls: représentant les liens de toutes les images associées à une storie.
- shared_user_ids, user_tags, folder_parent_names etc.

Recommandations :

Il serait intéressant de créer des tables correspondantes à certains attributs puis les associer aux tables avec des clés étrangères (relation one-to-many ou many-to many) ceci permettrait de normaliser les tables.

Exemple : comment_user_id deviendrait une table avec des colonnes : storie_hash, et user_id ayant pour clef primaire la combinaison (storie_hash , user_id) chacune étant une clé étrangère, ceci représenterait la relation many_to_many qu' il y a entre les tables stories et user_table

Bien que les colonnes calculées puissent être plus performantes pour des calculs simples et fréquemment utilisés, les nouvelles tables peuvent être plus adaptées pour des calculs complexes nécessitant des opérations de jointure. Il est souvent nécessaire de trouver un équilibre entre la simplicité, la performance et la normalisation.

2- Absence des clés étrangères explicites : il peut être plus difficile de garantir cette intégrité référentielle, ce qui signifie que les relations entre les tables sont maintenues de manière cohérente. Ceci peut entraîner des données orphelines ou des incohérences.

Sans clés étrangères, des anomalies de mise à jour peuvent survenir lorsqu'il y a des changements dans les données. Par exemple, si une ligne dans la table parentale est supprimée, cela peut conduire à des problèmes si les références dans la table fille ne sont pas correctement gérées.

La maintenance du schéma de la base de données peut être plus difficile sans clés étrangères, car il devient plus complexe de comprendre et de gérer les relations entre les tables.

Recommandations :

Définir et documenter explicitement les clés étrangères surtout dans le cadre d'une conception de base de données relationnelle ceci permettrait de garantir l'intégrité référentielle entre les tables et faciliter la maintenance.

3- Il existe certains attributs mal renommés comme (ng,nt,ps) présents dans les tables feeds et social_feeds qui ne sont ni claires ni explicites. Etant donné qu'il n'existe aucune clé étrangère ni dans la table feeds faisant référence à l' id de la table social_feed et vice-versa nous ne savons pas si les attributs représentent les mêmes choses ou pas .

Recommandations :

Donner des noms plus explicites et plus signifiant ceci faciliterait la compréhension de leur contenu.

Exemple :

- ng = negative_feed,
- nt = neutral_social_feed,
- ps = positive_value

Avantages liés au code source de manipulation de la base de données

1- Le code est bien structuré, il est bien réparti entre différents modules ce qui facilite l'évolution et la maintenance de code. (Fichiers : BlurDataBase.java, BlurDataBaseHelper.java, DatabaseConstants.java etc.)

2- Noms des variables assez explicites suivant une certaine convention de codage ce qui favorise la compréhension de leurs contenus

3- Il existe une quantité réduite de code dupliqué

4- Le code est compréhensif, ce qui favorise les modifications et maintenances en cas d'évolution de code

5- Le code de création des tables est isolé et séparé du code de manipulation des données de ces tables

Inconvénients liés au code source de manipulation de la base de données

1- Il existe des commentaires mais on remarque aussi que plusieurs fonctions ne sont pas spécifiées ou commentées.

Recommandations :

Spécifier toutes les fonctions : ajouter des commentaires permettant de mieux comprendre leurs fonctionnements.

2- Absence des tests Unitaires : cette absence pourrait affecter la qualité du code. L'absence de tests unitaires augmente le risque de bugs et d'erreurs logicielles. Les bugs peuvent être introduits lors du développement de nouvelles fonctionnalités, de modifications de code.

Recommandations :

Il faudrait tester chaque fonctionnalité de l'application avant de déployer l'application en production. La création des tests unitaires permettrait de rendre le code plus fiable et de se rassurer que chaque fonctionnalité fonctionne selon ses spécifications.