Consignes projet pour le cours « BdD et SQL »

Objectif : création d'une base de données

Livrables : chaque groupe doit remettre trois éléments

1. Un document décrivant la base de données telle qu'elle serait si vous aviez tout le temps nécessaire pour mener à bien la réalisation : objectif, intégration dans une application, périmètre des données, principaux traitements à effectuer (environ une page).
2. Une réalisation technique (maquette) permettant de mettre en œuvre **deux** des principaux traitements prévus.
3. Un descriptif de la réalisation technique ; ce descriptif doit comporter le modèle de données et une mode opératoire de test pour chacun des deux traitements mis en œuvre.

Exigences techniques pour la base de données maquette :

Votre maquette doit comporter au moins :

- 4 tables avec au moins deux lignes de données chacune.
- 1 vue
- 1 rôle pourvu des privilèges que devrait avoir un utilisateur standard

Ces chiffres sont bien sûr un minimum, vous pouvez faire plus si vous le souhaitez !

Calendrier :

- **Pour lundi 28 avril**, vous devez former une équipe de 5 étudiants maximum. Je formerai ensuite des groupes avec les éventuels étudiants non inscrits.
- **Pour mercredi 30 avril**, chaque groupe doit me soumettre pour validation le thème choisi pour sa base de données.
- **Lundi 26 mai à 12:00 :** limite pour la remise du projet final.

Modalités de remise :

- Dès que votre équipe est constituée, je vous attribue un nom d'équipe, un rôle (groupe) et une base de données vide sur mon serveur
- Vos documents doivent comporter le nom de votre équipe.
- Vous devez m'envoyer les documents sur antoine.dinimant@epita.fr avec l'ensemble des membres de l'équipe en copie.
- Votre maquette doit être réalisée ou déployée sur mon serveur MySQL.

Critères d' évaluation

- Bon fonctionnement de la maquette
- Ambition technique et sophistication
- Qualité de la réalisation et des documents

Exemple

Qu'est-ce qu'un traitement ?

Il s'agit de prendre un cas d'utilisation typique de votre base de données et de décrire ce que votre base doit réaliser.

Par exemple, lors de nos séances sur la base de données « trains et voyageurs », nous avons mis en place le traitement « vente d'un billet » selon le processus suivant :

- Données en entrée : ID de voyageur, ID de train, Prix
- S'il reste au moins une place sur le train, ajouter un billet dans la table Billet, décrémenter le nombre de places disponibles et envoyer un statut OK
- Sinon, ne rien faire et envoyer un statut d'erreur

Qu'est-ce qu'un mode opératoire ?

Le mode opératoire de test indique à un testeur ce qu'il doit faire pour vérifier que votre traitement fonctionne comme prévu. Vous devez donc y indiquer ce que la base de données **NE FAIT PAS** elle-même et qui doit être réalisé par le testeur.

- Situation initiale, éventuellement instructions de mise en situation de test.
- Comment déclencher chaque étape et/ou chaque scénario à tester.
- Quel est le résultat attendu.

Exemple : *mode op' de test du traitement « vente d'un billet » :*

Situation initiale :

- le train 4 doit exister et avoir une et une seule place libre .
- les voyageurs 6 et 7 doivent exister et ne pas avoir de billet sur le train 4.

Mise en situation initiale :

```
UPDATE Trains
SET NbPlaces = 1
WHERE IDtrain = 4 ;

DELETE FROM Billets
WHERE Idtrain = 4
  AND IDvoy IN (6, 7) ;
```

Scénario 1 : « il reste au moins une place »

```
CALL CreerBillet(7, 4, 25) ;
```

Résultat attendu :

- L'appel ci-dessus doit s'exécuter sans erreur
- Le billet doit apparaître dans la table des billets
- Le nombre de places du train doit être descendu à zéro

Scénario 2 : « il n'y plus de place »

Ne pas réinitialiser la situation.

```
CALL CreerBillet(6, 4, 25) ;
```

Résultat attendu :

- L'appel ci-dessus doit provoquer un message d'erreur indiquant la cause d'erreur.
- Les données des tables Trains et Billets ne doivent pas avoir changé.

Sécurité de votre projet

Rôles

- Pour mémoire, le modèle de sécurité de MySQL prévoit des utilisateurs et des rôles, mais pas de groupes. Ce sont les rôles qui vont servir de groupes.
- Chaque équipe dispose de deux rôles :
  - Un rôle développeur, sur le modèle **EquipeZ**, attribué aux étudiants de l'équipe.
  - Un rôle utilisateur, sur le modèle **role_user_z**, que l'équipe peut administrer (WITH ADMIN OPTION).
- Si vous pensez avoir besoin de rôles supplémentaires ou portant d'autres noms, demandez-moi de les créer.
- Les noms d'utilisateurs et de rôles sont sensibles à la casse, donc attention aux majuscules/minuscules !

Privilèges

- Le rôle développeur dispose de tous les privilèges sur le schéma attribué à l'équipe
- Tous les étudiants sont membres du rôle cyberb1 qui dispose de quelques privilèges système, en particulier CREATE USER.

Comment organiser la sécurité du projet ?

- Octroyez à votre rôle utilisateur les privilèges nécessaires pour qu'il puisse réaliser les traitements prévus, et eux seuls
- Créez un compte utilisateur, et attribuez-lui le rôle utilisateur
- Connectez-vous avec ce compte pour tester

Sécurité des différents objets

Tables

- Octroyez les privilèges nécessaires parmi SELECT, INSERT, UPDATE et DELETE au rôle utilisateur.
- SELECT et UPDATE peuvent être limités à certaines colonnes

```
GRANT SELECT, INSERT ON matable TO role_user_z ;
GRANT UPDATE (macolonne) ON matable TO role_user_z ;
```

Triggers

- Les triggers MySQL sont toujours exécutés avec les privilèges de leur créateur, mais sans prendre en compte les délégations.
- Vous devez donc indiquer le rôle développeur comme étant le créateur du trigger, car c'est lui qui a tous les privilèges :

```
CREATE DEFINER = EquipeZ TRIGGER montrigger
BEFORE INSERT ON matable…
```

Vues : vous avez deux stratégies :

- Si vous créez la vue pour éviter de donner trop de privilèges aux utilisateurs, la vue doit avoir les privilèges du développeur ; indiquez alors le rôle développeur comme créateur de la vue :

```
CREATE OR REPLACE DEFINER = EquipeZ VIEW mavue AS SELECT…
```

- Si au contraire vous voulez que la vue s'exécute avec les privilèges de l'utilisateur, précisez-le :

```
CREATE OR REPLACE SQL SECURITY INVOKER VIEW mavue AS SELECT…
```

- Octroyez les privilèges nécessaires parmi SELECT, INSERT, UPDATE et DELETE au rôle utilisateur, comme pour une table.

Fonctions et procédures : comme pour les vues, vous avez deux stratégies :

- Si vous voulez que la fonction ou procédure ait de meilleurs privilèges, indiquez le rôle développeur comme créateur :

```
CREATE DEFINER = EquipeZ FUNCTION mafonction() RETURNS…
```

- Si au contraire vous voulez que la vue s'exécute avec les privilèges de l'utilisateur, précisez-le :

```
CREATE FUNCTION mafonction RETURNS INT SQL SECURITY INVOKER BEGIN…
```

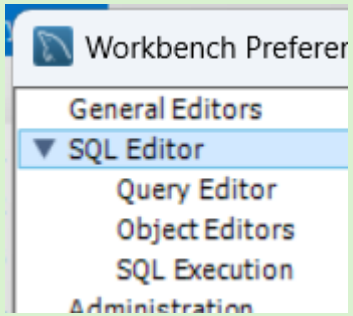
- Octroyez le privilège EXECUTE au rôle utilisateur en précisant le type d'objet :

```
GRANT EXECUTE ON FUNCTION mafonction TO role_user_z ;
GRANT EXECUTE ON PROCEDURE maprocédure TO role_user_z ;
```

Erreur 1175 avec le safe update mode

Qu'est-ce que c'est ?

- Le Safe Update Mode est une protection qui vous empêche de modifier (UPDATE) ou supprimer (DELETE) trop de lignes à la fois.
- Par défaut, MySQL Workbench active cette protection
- La définition exacte du « trop » est « il n'y a pas de filtre WHERE sur une colonne clef »



Comment régler Workbench pour qu'il n'active pas le safe update ?

- Dans Edit > Preferences, allez sur SQL Editor.
- Dans la section Other, décochez la case Safe Updates
- Cela sera valable pour toutes vos prochaines connexions, mais ne change rien pour la session en cours

Comment savoir si le safe update est activée dans ma session ?

- Exécutez la requête suivante :
  ```
  SELECT @@sql_safe_updates ;
  ```
- 1 veut dire que vous êtes « protégé » et 0 veut dire que le safe update est désactivé

Comment désactiver le safe update pour ma session en cours ?

- Exécutez la requête suivante :
  ```
  SET sql_safe_updates = FALSE ;
  ```
- Utilisez bien sûr TRUE si vous souhaitez le réactiver.