TD N°25 – BDD MOTO

I – Soit le MLD suivant :

Moto (num\_moto, nom-moto, cylindrée, catégorie)

Courir (num\_Moto#, num\_motard#, datecourse, place)

Motard (num\_motard, nom\_motard, num\_ecurie#)

Ecurie (num\_ecurie, nom\_ecurie, classement)

Vous n’avez pas les droits pour créer une base de données, ni pour créer des utilisateurs sous Mydon ni sous Matedon. Donc pour vous entrainer vous pouvez utiliser un Mysql en local (wampserver ?, phpMyadmin ?) mais là on va faire la partie « admin bdd » sur papier)

1) – Créer un utilisateur « Suzuky » dont le mot de passe est Kawa

2) – Donner à Suzuky le droit de créer, modifier, détruire une table, créer une vue et bien sûr de se connecter !

3) – Créer un utilisateur « Harley » dont le mot de passe est Harley

4) – Attribuer à cet utilisateur les droits : insertion sur Moto, motard et écurie, sélection sur toutes les tables et connexion.

5) – Quels utilisateurs auront le droit d’exécuter votre script de création de table ? Pourquoi ?

6) – Rajouter à la table motard la rubrique Année-de-naissance

7) – Ecrire la requête permettant de diminuer de 1 place le classement des motos APRILIA pour ne pas avoir respecté le règlement des courses

8) – Retirer à « Harley » tous les droits sauf celui de se connecter

9) – Créer un trigger qui vérifie qu’il ne peut pas exister 2 écuries ex æquo

10) – Créer une vue regroupant les courses des motos de l’écurie Happy puis donner le droit à Harley d’utiliser cette vue

11) – Ecrire une procédure stockée qui affiche la liste des motos « conduites » par un motard donné

# CORRECTION

1.

* Si vous êtes en « local » et que vous en avez les droits :
	+ Create database bddMoto;
* Pour visualiser la liste de toutes les bdd
	+ Show databases
* Puis se positionner dans cette bdd
	+ Use bddMoto

create user SUZUKY

identified by ‘KAWA’;

2.

grant connect to SUZUKY; -> seulement pour sql server

sous Sql Server :

grant create table to SUZUKY;

grant create view to SUZUKY;

ou grant create table, create view to SUZUKY;

sous Mysql : grant create on \*.\* to ' SUZUKY '

3.

create user Harley

identified by 'Harley';

sous SqlServer :

sp\_addlogin 'Harley', 'harley'

use db\_aaud;

sp\_adduser 'Harley', 'user\_harley'

4.

grant select, insert

on ecurie

to Harley;

grant select, insert

on Moto

to Harley;

grant select, insert

on motard

to Harley;

grant select

on courir

to Harley;

ou grant select, insert

on any table to Harley;

puis revoke insert

on courir from Harley;

grant connect to Harley; -> Seulement pour Sql Server

5.

Le script fonctionnera sous SUZUKY car il a le droit de créer des tables. En revanche, il pourra faire des insertions dans SES tables, s’il les crée mais pas sur MES tables car on ne lui a pas donné le droit

Le script ne fonctionnera pas sous Harley car l'utilisateur ne possède pas l'autorisation de créer des tables. Il a le droit d’insertion sur certaines tables, mais il faudra modifier le script pour préfixer les tables par le nom du propriétaire (moi).

6.

alter table motard

add Annéedenaissance Numeric(4);

alter table motard

add constraint c\_dat check (Annéedenaissance like 19\_\_ or Annéedenaissance like 20\_\_);

7.

update ecurie

set classement = classement + 1 Attention au + 1 et non - 1

where nomecurie = 'aprilia';

Pour vérifier :

select \* from ecurie

order by classement

8.

revoke select

on courir

from Harley;

revoke select, insert

on moto

from Harley;

revoke select, insert

on motard

from Harley;

revoke select, insert

on ecurie

from Harley;

Ou

Revoke all on ecurie from Harley

Revoke all on courir from Harley

Revoke all on moto from Harley

Revoke all on motard from Harley

-> On pourrait utiliser Deny au lieu de Revoke si on veut être sûr que personne ne lui transfère les droits.

9.

Si on interdit et qu’on inverse les classements …

Ce n’est pas idéal car là on fait +1 … mais on peut imaginer +2, +3 …

Donc on interdit et on affiche un message erreur

Sous SQL Server

create trigger T\_verif\_class ON ecurie FOR insert, update

as

declare @tmp as smallint

select @tmp = count(\*) from ecurie where classement = (select classement from inserted)

if @tmp <> 1

begin

 rollback

 select 'impossible'

end

pour vérifier la présence du trigger

sp\_helptrigger ecurie



pour vérifier que le trigger fonctionne

select \* from ecurie



Puis on peut tester en changeant le classemement pour aprilia puis pour honda

update ecurie

set classement = classement + 1

where lower(nomecurie) = 'aprilia';

-> ne fonctionne pas

update ecurie

set classement = classement + 1

where lower(nomecurie) = 'honda';

-> fonctionne

Sous Mysql

On ne peut pas utiliser le commit ou le rollback dans un trigger

delimiter $$

create trigger T\_verif\_class BEFORE update, insert ON ecurie FOR EACH ROW

BEGIN

declare tmp int;

DECLARE msg VARCHAR(255);

select count(\*) into tmp from ecurie where classement = New.classement;

if (tmp > 0) then

 set msg = "DIE: non non non, vous ne pouvez pas avoir deux écuries ex-eaquo...";

 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = msg;

end if;

END;

10.

* Créer une vue regroupant les courses des motos de l’écurie Happy

CREATE VIEW V\_courses\_Happy

AS

SELECT c.\* FROM courir c, ecurie e, motard m

WHERE c.nummotard = m.nummotard

AND m.numecurie = e.numecurie

AND nomecurie = ‘Happy’

Ou

Create view V\_courses\_Happy

As

Select \*

From courir

Where nummotard IN

 (Select nummotard

 from motard

 Where numecurie =

 (Select numecurie

 From ecurie

 Where nomecurie = 'Happy'))

* Select \* from V\_courses\_Happy
* Donner le droit à Harley d’utiliser cette vue

Grant select on V\_courses\_Happy to Harley ;

11) – Ecrire une procédure stockée sous MySql qui affiche la liste des motos « conduites » par un motard donné

delimiter $$

create procedure ListeMotoMess (in nom char(20))

begin

declare tmp int;

declare tmp2 int;

set tmp = 0;

set tmp2 = 0;

select nummotard into tmp

from motard

where nommotard = nom ;

select count(m.nummoto) into tmp2 from motard mo, moto m, courir c

 where c.nummoto = m.nummoto

 and c.nummotard = mo.nummotard

 and nommotard = nom ;

if tmp = 0 then

select 'pas de motard avec ce nom';

elseif tmp2 = 0 then

 select 'ce motard n''a pas de course';

else select m.\* from motard mo, moto m, courir c

where c.nummoto = m.nummoto

and c.nummotard = mo.nummotard

and nommotard = nom ;

end if;

end $$

delimiter ;

puis tester avec :

call ListeMotoMess('Motokiller')



call ListeMotoMess(' LORENZO ')



call ListeMotoMess(' TOTO ')



call ListeMotoMess('Cross')



Si c’était une épreuve d’examen à E6 voici un barème approximatif :

 en tête 3

 declaration variables / call 2

 test motard existe 3

 message erreur 2

 test moto existe (course) 3

 message erreur 2

 afficher liste moto 3

 logique / synthaxe 2