Bases de données
PL-SQL
Procédures et fonctions stockées sous MySQL
CORRIGE du TP - 1

Bertrand LIAUDET

1. La bibliothèque

1) Créer la BD « biblio » à partir du script fourni.
2) Ecrire une fonction qui calcule, pour un adhérent donné, le nombre de jours restant avant d'être en retard. Si l'adhérent n'a pas d'emprunts en cours, on renvoie NULL. Si l'adhérent est en retard, on renvoie un résultat négatif. On prendra en compte la possibilité d'avoir des emprunts avec des dureeMax différentes et des emprunts en cours avec des dates d'emprunt différentes. Dans ce cas, on renverra le nombre de jours restant le plus petit et le nombre de jours de retard le plus grand.

Analyse de la fonction :

on projette le nombre de jours restants par emprunt : c'est un attribut calculé

```
Select *, dureeMax –to_days(current_date) + to_days(dateEmp) as
reste
from emprunter
where dateRet is null
order by na ;
```

on récupère le min par adhérent

```
Select na, min(dureeMax –to_days(current_date) +
to_days(dateEmp)) as resteMin
from emprunter
where dateRet is null
group by na ;
```

On en déduit la fonction

```
drop function if exists nbjRestants;
delimiter //
create function nbjRestants(my_na int)
     returns int
     deterministic
begin
     declare res int;
     Select min(dureeMax–to_days(current_date)+to_days(dateEmp))
     Into res
     from emprunter
     where dateRet is null and na=my_na ;

     return res ;
```

```
end ;
//
delimiter ;
```

3) Utiliser cette fonction pour afficher la situation de tous les adhérents.

```
Select *, nbjRestants(na) from adherents ;
```

4) Ecrire une procédure qui permette de lister les emprunts d'un adhérent identifié par son numéro.

```
drop procedure if exists mesEmprunts;
delimiter //

create procedure mesEmprunts(my_na integer)
begin
     Select * from emprunter where na=my_na order by dateEmp;
end ;
//
delimiter ;

call mesemprunts(1) ;
```

5) Ecrire une procédure qui affiche les exemplaires disponibles d'un titre (on fera une version OUTER JOIN et une version NOT IN). Pour se faciliter la tâche, on a intérêt à d'abord traiter la question : « combien y a-t-il exemplaires disponibles du titre 'NARCISSE ET GOLDMUND' » avec les deux versions demandées, pour ensuite passer à l'écriture de la procédure stockée.

Version OUTER JOIN

```
Select l.*, o.*
From emprunter e
Right Join livres l on (l.nl=e.nl and e.dateRet is null)
Join oeuvres o using (no)
Where  upper(o.titre) ='NARCISSE ET GOLDMUND'
And e.nl is null;
```

Version NOT IN

```
Select l.*, o.*
From livres l
Join oeuvres o using (no)
Where  upper(o.titre) ='NARCISSE ET GOLDMUND'
And l.nl Not in(
    Select l.nl
    From emprunter e
    Join livres l on (l.nl=e.nl and e.dateRet is null)
    Join oeuvres o using (no)
    Where  upper(o.titre) ='NARCISSE ET GOLDMUND'
);
```

```
drop procedure if exists dispo;
delimiter //

create procedure dispo(my_titre varchar(150))
begin
     Select l.*, o.*
     From emprunter e
     Right Join livres l on (l.nl=e.nl and e.dateRet is null)
     Join oeuvres o using (no)
```

```
        Where  upper(o.titre) =my_titre
        And e.nl is null;
    end ;
    //
    delimiter ;

    call dispo('NARCISSE ET GOLDMUND') ;
```

6) Ecrire une procédure qui affiche les titres d'un auteur et le nombre d'exemplaires disponibles par titre. On testera avec LEWIS CAROLL, GILBERT HOTTOIS et kenneth white. Pour se faciliter la tâche, on a intérêt à commencer par traiter la question : « Les exemplaires dispo de Lewis Caroll » puis « Le nombre d'exemplaires dispo par titre de Lewis Caroll » pour enfin écrire la procédure stockée.

Les exemplaires dispo de Lewis Caroll

```
Select l.*, o.*
From emprunter e
Right Join livres l on (l.nl=e.nl and e.dateRet is null)
Join oeuvres o using (no)
Where  upper(o.auteur) ='LEWIS CAROLL'
And e.nl is null;
```

Le nombre d'exemplaires dispo par titre de Lewis Caroll

```
Select o.no, o.titre, count(*)
From emprunter e
Right Join livres l on (l.nl=e.nl and e.dateRet is null)
Join oeuvres o using (no)
Where  upper(o.auteur) ='LEWIS CAROLL'
And e.nl is null
Group by o.no, o.titre;
```

LA PROCEDURE :

```
    drop procedure if exists dispoAuteur;
    delimiter //

    create procedure dispoAuteur(my_auteur varchar(100))
    begin
        Select o.no, o.titre, count(*)
        From emprunter e
        Right Join livres l on (l.nl=e.nl and e.dateRet is null)
        Join oeuvres o using (no)
        Where  upper(o.auteur) =upper(my_auteur)
        And e.nl is null
        Group by o.no, o.titre;
    end ;
    //
    delimiter ;

    call dispoAuteur('LEWIS CAROLL') ;
    call dispoAuteur('GILBERT HOTTOIS') ;
    call dispoAuteur('kenneth white') ;
```

7) Ecrire une procédure qui permette d'enregistrer un emprunt.

```
drop procedure if exists emprunter;
delimiter //
create procedure emprunter (v_nl integer, v_na integer, v_dateEmp
   date, v_dureeMax integer)
begin
```

```
    insert into emprunter values(v_nl, v_dateEmp, d_max, NULL,
    v_na);
end;
//
delimiter ;
```

8) Modifier la table des emprunts : mettez la valeur par défaut de la durée max à 14.

```
alter table emprunter modify dureemax integer not null default 14;
```

9) Ecrire une nouvelle procédure qui enregistre un emprunt et gère tous les cas d'erreur (le livre n'existe pas, l'adhérent n'existe pas, le livre est déjà emprunté, l'adhérent emprunte déjà 3 livres, etc.). La procédure impose la date du jour comme date d'emprunt. La procédure renverra un numéro de code pour chaque erreur.

On se donne une fonction qui dit si un livre est disponible ou pas :

```
drop function if exists estDispo;
delimiter //
create function estDispo(my_nl int)
     returns int
     deterministic
begin
     declare res int;
     Select count(*)
     Into res
     from emprunter
     where dateRet is null and nl=my_nl ;
     return not res ;
end ;
//
delimiter ;
```

On se donne une fonction qui renvoie le nombre de livres actuellement empruntés par un adhérent :

```
drop function if exists nbEmprunts;
delimiter //
create function nbEmprunts (my_na int)
     returns int
     deterministic
begin
     declare res int;
     Select count(*) into res
     from emprunter
     where dateRet is null and na=my_na ;
     return res ;
end ;
//
delimiter ;
```

PROCEDURE : solution 1

```
drop procedure if exists emprunterLA;
delimiter //
create procedure emprunterLA (v_nl integer, v_na integer)
begin
    if (select nl from livres where nl=v_nl) is null then
         select concat('Le livre n° ', v_nl,' n\'existe pas') as
    erreur;
    elseif (select na from adherents where na=v_na) is null then
         select concat('L\'adhérent n° ', v_nl,' n\'existe pas') as
    erreur;
```

```
    elseif estDispo(v_nl)=0 then
        select concat('Le livre n° ', v_nl,' n\'est pas dispo') as
    erreur;
    elseif nbEmprunts(v_na)=3 then
        select concat('L\'adhérent n° ', v_nl,' a déjà 3 livres
    empruntés') as erreur;
    else
        insert into emprunter (nl, dateEmp, na) values(v_nl,
    current_date(), v_na);
    end if;
end;
//
delimiter ;
```

PROCEDURE : solution 2 : on sépare le calcul de l'IHM

Fonction de calcul :

```
drop function if exists emprunterLA;
delimiter //
create function emprunterLA (v_nl integer, v_na integer)
    returns int
    deterministic
begin
    -- vérification de l'existence du livre
    if (select nl from livres where nl=v_nl) is null then
        return 1;
    -- vérification de l'existence de l'adhérent
    elseif (select na from adherents where na=v_na) is null then
        return 2;
    -- vérification de la disponibilité du livre
    elseif estDispo(v_nl)=0 then
        return 3 ;
    -- vérification du nombre d'emprunts de l'adhérent
    elseif nbEmprunts(v_na)=3 then
        return 4 ;
    -- si aucun problème : on renvoie 0
    else
        return 0;
    end if;
end;
//
delimiter ;
```

PROCEDURE d'IHM

```
drop procedure if exists emprunterLA;
delimiter //
create procedure emprunterLA (v_nl integer, v_na integer)
begin
    declare res int;
    set res=emprunterLA(v_nl, v_na);
    if res=1 then
        select concat('Le livre n° ', v_nl,' n\'existe pas') as
    erreur;
    elseif res=2 then
        select concat('L\'adhérent n° ', v_nl,' n\'existe pas') as
    erreur;
    elseif res=3 then
        select concat('Le livre n° ', v_nl,' n\'est pas dispo') as
    erreur;
    elseif res=4 then
        select concat('L\'adhérent n° ', v_nl,' a déjà 3 livres
    empruntés') as erreur;
    elseif res=0 then
```

```
        insert into emprunter (nl, dateEmp, na) values(v_nl,
    current_date(), v_na);
    end if;
end;
//
delimiter ;
```

10) Rajouter le trigger qui permet de gérer l'attribut « emprunté », booléen permettant de savoir si un livre est emprunté ou pas (cf. TP précédent).

```
alter table livres add libre integer default 1 ;

update livres set libre=1 ;

update livres set libre=0
where nl=any(select nl from emprunter where dateret is null);
```

```
drop trigger if exists tai_emprunter;
# tai : trigger after insert
delimiter //
create trigger tai_emprunter
after insert on emprunter
for each row
begin
    if new.dateret is null then
        update livres set libre=0 where nl=new.nl;
        # select new.nl into @vnl; si on veut verifier la valeur de
    new.nl
    end if ;
end;
//
delimiter ;
```

```
drop trigger if exists tau_emprunter;
# tau : trigger after update
delimiter //
create trigger tau_emprunter
after update on emprunter
for each row
begin
    if new.dateret is not null then
        update livres set libre=1 where nl=new.nl;
        # select new.nl into @vnl; pour verifier la valeur de
    new.nl
    end if;
end;
//
delimiter ;
```

11) Vérifier que ce trigger fonctionne avec la procédure stockée de l'exercice précédent.

```
Select estDispo(15) ;

Select nbEmpruntes(2) ;

Select * from livres where nl=15 ;

Call emprunterLA(15, 2) ;

Select * from livres where nl=15 ;
```

12) Créer la procédure stockée qui permette d'enregistrer un retour en gérant tous les cas d'erreur (le livre n'existe pas, l'adhérent n'existe pas, le livre n'est pas emprunté, etc.) et en imposant la date du jour comme date de retour. La procédure renverra un numéro de code pour chaque erreur et le nombre de jour de retard s'il y a lieu.