Série N 4

Module 18 : Système de Gestion de Base de Données (II)

FILIERE : TDI
NIVEAU : 2éme année

Exercices 1:

Sur le schéma relationnel (GestStg) suivant :

Stagiaire (idstg, nom, moyenne)
Module (idmod, libelle, coeff)
Note (idstg, idmod, note)

Questions :

1) Définir l'intégrité de domaine et référentiel avec des triggers.
2) Implémenter la suppression en cascade avec les triggers.
3) Recalculer la moyenne pour chaque modification dans les notes

Exercices 2:

Soit le schéma relationnel « Agence ».

Station (**nomStation**, capacité, lieu, région, tarif)
Activite (***nomStation***, **libellé**, prix)
Client (**id**, nom, prénom, ville, région, solde)
Sejour (***idClient***, ***station***, **début**, nbPlaces)

Station

Nomstation	**capacité**	**lieu**	**région**	**tarif**
Venus	350	Guadeloupe	Antilles	1200

Activité

nomstation	**libellé**	**prix**
Venus	Voile	150
Venus	Plongée	120

Client

ID	**nom**	**prénom**	**ville**	**région**	**solde**
10	Gogg	Philipes	Londres	Europe	1246.5
20	Pascal	Blais	Paris	Europe	6763
30	Kerouac	Jack	New York	Amérique	9812

Séjour

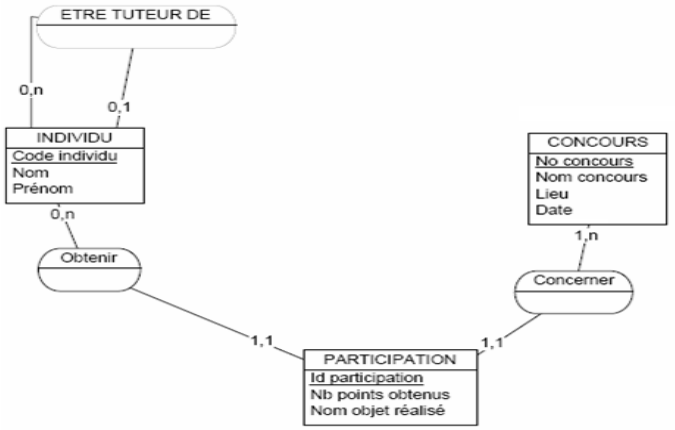
idClient	**station**	**début**	**Nbplaces**
20	Venus	03/08/2003	4

Questions :

1) Créer une PS *NomClient* qui prend en entrée l'id d'un client et qui affiche une chaîne contenant le prénom et le nom du client .
2) Créer une fonction *Activités* qui prend en entrée le nom du station et produit une chaîne de caractères contenant l'énumération des activités de la station (par exemple, "`Ski`, `Yoga`, `Massage`").
3) Créer ensuite une vue qui affiche les stations, avec un attribut supplémentaire donnant la liste des activités (par appel à la fonction bien sûr).
4) Créer une PS *Actualiser* qui prend en entrée un pourcentage et le nom d'une station, et augmente le tarif de la station et le prix de chacune de ses activités du pourcentage indiqué.
5) Implantez par un trigger la règle suivante : si le prix d'une activité baisse, alors le tarif de la station doit augmenter de la différence. Indication : le trigger doit se déclencher sur une modification, et doit faire un UPDATE de la station pour ajouter la différence entre l'ancienne et la nouvelle valeur. (faites le teste)
6) On veut disposer de l'information nbActivites dans la table Station. Pour cela :
   (a) Ajoutez la colonne nbActivites avec pour valeur par défaut 0.
   (b) Créez un trigger qui maintient cette information.
7) Interdisez par un trigger l'insertion d'une ligne dans la table Séjour si le solde du client est inférieur au nombre de places multiplié par le tarif de la station.

Exercices 3:

On propose dans ce qui suit de créer une base de données de gestion des concours.



Créer la base de données correspondante au MCD ci-dessus.
Créer des déclencheurs qui permettent de:

1) Lors de la suppression d'un individu, on supprime son tuteur s'il n'est pas tuteur d'un autre individu.
2) Interdire de modifier le code d'un individu déjà ajouté.
3) Ajouter le champ [NB points total] à la table Individu et écrire tout trigger possible de le maintenir à jour
4) Interdire un individu à participer à des concours plus de quatre fois
5) Interdire un individu à participer au même concours plus de deux fois
6) Lors de l'insertion d'un enregistrement 'concours' on change le jour de la date par un, et l'heure par 00 :00 :00.000

Exercices 4 :

Soit une base de données de gestion des projets qui contient les relations suivantes:

Employe (Matr, NomE, Grade, DatEmb, Salaire, Commission, NDept)
Département (NumDept, NomDept, Lieu)
Projet (CodeP, NomP, PrevisionH)
Participation (MatrEmp, CdeP, Heures)

1) Créer une procédure stockée qui insère une nouvelle participation et qui Affiche le nombre total d'heures des participations pour le projet concerné par la participation ajoutée et le nombre de participation de l'employé concerné par cette participation et son salaire avec commission.
2) Créer une fonction qui prend le Code projet et retourne le niveau de réalisation en % par rapport à la prévision.
3) Créer une fonction qui prend le Matricule Employé et retourne une table (NomP, NiveauR), les projets dans les quelles il a participé.
4) Créer un (des) trigger(s) qui supprime (nt) en cascade après la suppression d'un département.

Correction :

Exercices 1 :

```
Use Master
--Drop database GestStg
Create Database GestStg
Go

Use GestStg
create table Stagiaire(IdStg int identity primary key, Nom varchar(20), Moyenne real)
Create table Matiere(IdMat int identity primary key, Libelle varchar(20), Coeff real)
Create table Note(IdStg int, IdMat int , Note real)
Go

Insert into stagiaire values ('Ali',null)
insert into stagiaire values ('Ahmed', null)
select * from stagiaire
delete from stagiaire where idstg=1
Go
Insert into Matiere values ('SGBD1',2)
Insert into Matiere values ('SGBD2',3)
select * from Matiere
Go
Insert into Note values (3,1,10)
Insert into Note values (3,2,11)
Insert into Note values (1,2,15)
Insert into Note values (2,2,11)
Insert into Note values (2,2,-5)
Insert into Note values (2,1,16)
delete from note where note=11 and idstg=3
Select * from Note
Go


-- 1) Créer un trigger qui affiche les lignes inséré pour insert sur la table
Matière.

Create trigger Aff on Matiere for insert
as
 select * from inserted
Go

-- 2) Refaire l'implémentation des contraintes d'intégrité suivant par des triggers:
--- Référentiel (clé étranger)
--- De domaine (0<=Note<=20)

Create trigger IntgRef on Note for insert,update
as
Begin
  if not exists(select * from stagiaire,inserted, matiere where
stagiaire.idstg=inserted.idstg and inserted.idmat=matiere.idmat) or not exists(select
* from inserted where note between 0 and 20)
    Begin
       RAISERROR('Violation d''integrité référentiel et de domaine',15,1)
       Rollback tran
    End
End
Go

-- 3) Implémenté la suppression en cascade

Create trigger SuppStg on stagiaire for delete
as
      delete from note where idstg in (select idstg from deleted)
```

```
Go
Create trigger SuppMat on Matiere for delete
as
      delete from note where idmat in (select idmat from deleted)
Go

-- 4)Recalculer la moyenne pour chaque modification dans les notes

Alter trigger IntgRef on Note for insert,update
as
Begin
  if not exists(select * from stagiaire,inserted, matiere where
stagiaire.idstg=inserted.idstg and inserted.idmat=matiere.idmat) or not exists(select
* from inserted where note between 0 and 20)
    Begin
       RAISERROR('Violation d''integrité référentiel et de domaine',15,1)
       Rollback tran
    End
  Update stagiaire set Moyenne=(select sum(note*coeff)/sum(coeff)
                                from matiere,note
                                where matiere.idmat=note.idmat
                                and note.idstg=stagiaire.idstg)
  Where stagiaire.idstg in (select idstg from inserted)
End
Go

Create trigger CalMoy on note for delete
as
  Update stagiaire set Moyenne=(select sum(note*coeff)/sum(coeff)
                                from matiere,note
                                where matiere.idmat=note.idmat
                                and note.idstg=stagiaire.idstg)
  Where stagiaire.idstg in (select idstg from deleted)
```

Exercices 2 :

```
Use Master
Create Database SGBD2_S4_Ex2
Go

Use SGBD2_S4_Ex2
Create table Station (nomStation varchar(30) primary key, capacité int, lieu
varchar(50), region varchar(50), tarif real)
Create table Activite (nomStation varchar(30) foreign key references Station
(nomStation), libelle varchar(100), prix real)
Create table Client (id int primary key, nom varchar(30), prenom varchar(30), ville
varchar(50), region varchar(50), solde real)
Create table Sejour (idClient int foreign key references Client (id), station
varchar(30) foreign key references Station (nomStation), debut smalldatetime,
nbPlaces int)
Go

Insert into station values ('Venus',350,'Guadeloupe', 'Antilles', 1200)
Go
Insert into activite values ('Venus','Voile', 150)
Insert into activite values ('Venus', 'Plongée', 120)
Go
Insert into Client values (10, 'Gogg', 'Philipes', 'Londres', 'Europe', 1246.5)
Insert into Client values (20, 'Pascal', 'Blais', 'Paris', 'Europe', 6763)
Insert into Client values (30, 'Kerouac', 'Jack', 'New York', 'Amérique', 9812)
Go
Insert into Sejour values (20, 'Venus', '03/08/2003', 4)
Go
```

```
--1) Créer une PS NomClient qui prend en entrée l'id d'un client et qui affiche une
chaîne contenant le
--- prénom et le nom du client .
Use SGBD2_S4_Ex2
Go
create proc NompCli @id int
as
  select nom + ' ' + prenom from client where id=@id
Go

-- 2) Créer une fonction Activités qui prend en entrée le nom du station et produit
une chaîne de
--- caractères contenant l'énumération des activités de la station (par exemple,
"Ski, Yoga,
--- Massage").

Create Function ListeActivite(@nstation varchar(30)) returns varchar(200)
as
begin
  declare @lib as varchar(30), @r varchar(200)
  set @r=''
  declare Lact cursor for select libelle from activite where nomStation like
@nstation
  open lact
  fetch next from lact into @lib
  while @@fetch_status=0
    Begin
      set @r=@r + @lib +', '
      fetch next from lact into @lib
    End
  close lact
  deallocate lact
  return @r
End
Go

select dbo.ListeActivite('venus')

-- 3) Créer ensuite une vue qui affiche les stations, avec un attribut supplémentaire
donnant la liste des
--- activités (par appel à la fonction bien sûr).
Create view ListStation
as
 select *, dbo.ListeActivite(nomstation) as 'Liste Activite' from station

-- 4) Créer une PS Actualiser qui prend en entrée un pourcentage et le nom d'une
station,
--et augmente le tarif de la station et le prix de chacune de ses activités du
pourcentage indiqué.

/* on utilisera les transaction pour que les deux instructions soit executé
*/
Create Proc Actualiser @pc real, @nstation varchar(50)
as
Begin
 begin tran
 begin try
   update station set tarif=tarif*(1+@pc/100) where nomstation like @nstation
   update activite set prix=prix*(1+@pc/100) where nomStation like @nstation
   commit tran
 end try
 begin catch
   declare @err varchar(200)
   select @err=ERROR_MESSAGE()
```

```
   raiserror(@err,15,1)
   rollback tran
 end catch
End

select * from station where nomstation like 'venus'
select * from activite where nomstation like 'venus'
exec Actualiser 10,'venus'

/*5) Implantez par un trigger la règle suivante : si le prix d'une activité baisse,
alors le tarif de la station doit augmenter de la différence. Indication :
le trigger doit se déclencher sur une modification, et doit faire un UPDATE
de la station pour ajouter la différence entre l'ancienne et la nouvelle valeur.
(faites le teste)*/

Create trigger Quest5 on activite for update
as
Begin
if (select ol.prix-nv.prix from inserted nv, deleted ol where nv.libelle=ol.libelle
                          and nv.nomstation=ol.nomstation)>0
  Begin
      update station set tarif=tarif+(select ol.prix-nv.prix from inserted nv,
deleted ol
                          where nv.libelle=ol.libelle
                          and nv.nomstation=ol.nomstation and
nv.nomstation=station.nomstation)
  End
End
Go

select * from station
select * from activite

-- pour une augmentation rien ne se passe
update activite set prix = 170 where nomstation like 'venus' and libelle like 'voile'
-- pour une diminution, il y as augmentation dans tarif du station
update activite set prix = 130 where nomstation like 'venus' and libelle like 'voile'

/* 6) On veut disposer de l'information nbActivites dans la table Station. Pour cela
:
(a) Ajoutez la colonne nbActivites avec pour valeur par défaut 0.
(b) Créez un trigger qui maintient cette information.
*/
-- a)
alter table station add nbActivites int default 0
update station set nbActivites=0

-- b)
Create trigger NbrAct on activite for insert, update, delete
as
 update station set nbActivites=(select count(*) from activite
                                 where activite.nomstation=station.nomstation)
 where nomstation in (select nomstation from inserted union select nomstation from
deleted)

select * from station
select * from activite

insert into activite values ('Venus','Sky',100)

/* 7)Interdisez par un trigger l'insertion d'une ligne dans la table Séjour
si le solde du client est inférieur au nombre de places multiplié par le tarif de la
station.
*/
```

```
Create trigger Quest7 on sejour for insert
as
Begin
 if (select solde-(nbPlaces*tarif) from client,inserted, station
     where client.id=inserted.idclient and inserted.station=station.nomstation)<0
   Begin
     Raiserror('Solde insiffusent !!',15,1)
     rollback tran
   End
End

select * from client
select * from sejour
select * from station
insert into sejour values (10,'Venus','30/1/2012',3)
```