Contrôle BDD – 7 déc. 2010

Nom: Prénom:

Récupérez le script de création de la base de données "Gestion Notes" à l'adresse suivante:

http://www.fil.univ-lille1.fr/~bilasco/bdd/dsm/notes.sql

Cette base de données contient trois tables appelées *notes, etud, cours*.

La table *notes* a les colonnes suivantes :
- **numet** - numéro de l'étudiant
- note - la note dans une matière
- **codmat** - le code à 3 lettres de la matière
- coeff - le coefficient de la matière.

La table *etud* a les colonnes suivantes :
- **numet** - numéro de l'étudiant
- nom - nom de l'étudiant
- prenom - nom de l'étudiant

La table *matières* a les colonnes suivantes :
- nomprof - nom du professeur (un professeur peut enseigner plusieurs matières)
- **codmat** – le code à 3 lettres de la matière (une matière est enseignée par un seul prof)

A la fin de l'examen, vous me rendrez cette feuille. Vous déposerez également sur l'application PROF, un fichier .txt portant votre nom et contenant l'ensemble des requêtes utilisées.

Contraintes d'intégrités (3 pts)

Question 1.1. Ecrivez une requête pour vérifier la validité des contraintes d'intégrité référentielle entre les tables *notes*, *etud* et *matières*.

select codmat from notes where codmat not in (select codmat from matieres) group by codmat

select numet from notes where numet not in (select numet from etud) group by numet

Question 1.2. Corrigez les éventuelles erreurs (si possible à l'aide des instructions UPDATE/DELETE selon le cas) et ajouter les contraintes référentielles dans la base. N'oubliez pas que pour ajouter les contraintes d'intégrité référentielle il faut tout d'abord identifier les clés primaires.

A défaut, vous pouvez modifier directement le ficher .sql - mais cela vous rapportera naturellement moins de points.

update notes set codmat='BDD' where codmat='BD'
update notes set codmat='ENG1' where codmat='ANG1'
delete from notes where numet not in (select numet from etud)

alter table matieres add constraint pk_mat primary key (codmat);
alter table notes add constraint fk_mat foreign key (codmat) references matieres(codmat);
alter table etud add constraint pk_etud primary key (numet);
alter table notes add constraint fk_etud foreign key (numet) references etud(numet);

Question 1.3. Testez les contraintes en proposant suffisamment des requêtes pour en juger le bon fonctionnement.

Dépendances fonctionnelles (3 pts)

Question 2.1. Parmi les DF de la table *notes* on observe que codmat -> coeff.
Ecrivez une requête pour vérifier le respect de la DF pour la matière BDD.
Combien de coefficients distincts doit-on trouver pour une matière pour que la DF soit respectée?
.....
.....

Question 2.2. Ecrivez une requête pour vérifier le validité de la DF pour tous les matières dans la base.
Select codmat from notes group by codmat having count(distinct coeff)>1

Question 2.3. Modifiez les enregistrements qui ne respectent pas la DF. Utilisez une requête UPDATE. On considère que la première valeur rencontrée est celle à retenir pour le reste (fiez-vous à l'ordre des enregistrements obtenues lors d'un SELECT).

update notes set coeff=1 where codmat='ENG1'
update notes set coeff=2 where codmat='BDD'

A défaut, vous pouvez modifier directement le ficher .sql - mais cela vous rapportera naturellement moins de points.

Requêtes diverses (6 pts)

Question 3.1. Calculez la moyenne, la plus petite et la plus grande note accordées aux étudiants par matière.
select codmat,avg(note),min(note),max(note) from notes group by codmat

Question 3.2. Calculez, par enseignant et par matière, la moyenne, la plus petite et la plus grande note accordées aux étudiants.
select codmat,avg(note),min(note),max(note) from notes natural join matieres group by nomprof,codmat

Question 3.3. Trouvez, par enseignant, l'étudiant auquel il a accordé la plus petite note.
select nomprof,nom from etud natural join notes natural join matieres m1 where note in (select min(note) from notes natural join matieres m2 where m2.nomprof=m1.nomprof)

Question 3.4. Affichez les noms de profs enseignant au moins deux matières dont au mois une où les étudiants ont des notes.
select nomprof from matieres group by nomprof having (count(distinct matieres.codmat)>1)
intersect
select nomprof from matieres where codmat in (select codmat from notes)

Question 3.5. Afficher le numéro d'étudiant, le nom et le prénom des étudiants n'ayant pas de note pour l'ensemble des matières présentes dans la table *notes*.
select numet, nom, prenom from etud natural join notes group by numet,nom,prenom having count(codmat)<(select count(distinct codmat) from notes)

Question 3.6. Affichez les noms de profs qui ont au moins un cours où tous les étudiants ont une note.
select distinct nomprof from matieres where not exists (select numet from etud where numet not in (select numet from notes where notes.codmat=matieres.codmat))

PL/SQL (5 pts) (si vous rencontrez des difficultés liées au PL/SQL, commencez par écrire les requêtes correspondantes … pour moins de points biensûr)

Question 4.1. Créez une fonction PL/SQL qui calcule la moyenne d'un étudiant dont le numéro étudiant est indiqué en paramètre.

create or replace function moy(num number) return number is
moy number;
begin
select avg(note*coeff) into moy from notes where numet=num;

```
return moy;
end;
```

Question 4.2. Créez une fonction PL/SQL qui renvoie le nombre de matières où un étudiant n'a pas eu de note, par rapport à l'ensemble des matières référencées dans la table *notes*

```
create or replace function manquantes(num number) return number is
nb number;
nbtot number;
begin
select count(distinct codmat) into nbtot from notes;
select count(distinct codmat) into nb from notes where numet=num;
return nbtot-nb;
end;
```

Question 4.3. Créez une procédure qui affiche le numéro d'étudiant, le nom et le prénom des étudiants recalés (ayant une moyenne <10) ou n'ayant pas l'ensemble des notes. Pour chaque étudiant vous indiquerez s'il n'a pas eu la moyenne ou s'il n'a pas passé tous les examens (par rapport au matières dans la table *notes*). *Bonus, à faire en dernier:* dans le cas ou il n'a pas passé tous les examens afficher les noms de matières manquantes

```
create or replace procedure pb is
cursor c is select nom, prenom, moy(numet) as m, manquantes(numet) as n from notes natural join etud group by numet, nom,prenom;
begin
for c1 in c loop
if c1.n<>0 or c1.m<10 then
dbms_output.put(c1.nom);
dbms_output.put(' ');
dbms_output.put(c1.prenom);
end if;

if c1.n=0 then
  if c1.m < 10 then
    dbms_output.put(' n a pas eu la moyenne ');
   dbms_output.put_line(c1.m);
  end if;
else
   dbms_output.put(' il lui manque ');
   dbms_output.put(c1.n);
   dbms_output.put_line(' matieres ');
end if;
end loop;
end;
```

Question 4.4. Créez une fonction qui vérifie la validité d'un nouveau enregistrement dans la table *etud.*

peut_on_inserer_note(numetud,codmat,coeff,note) : number

Vérifiez

- que la note saisie est comprise entre 0 et 20,
- que le coefficient (supérieur à zéro) est le même pour les notes déjà saisies dans la même matière.

Un code retour inférieur à 0 est renvoyé en cas de problème, la valeur 1 est renvoyée sinon.

```
Create or replace function peut_on_inserer_note(numetud number, codmatiere notes.codmat%type, coeff notes.coeff%type, note notes.note%type) return number is
begin

if note < 1 or note > 20 then
  return -1;
end if;

if coeff <0 then
   return -2
end if;
```

```
select count(codmat) into nb_mat from notes where codMat=codmatiere;
if nb_mat >0 then
  select coeff into coeff_ex from notes where codmat=codmatiere group by coeff;
  if coeff_ex <> coeff then
     return -3;
  end if;
end if;
return 0;
end;
```

En absence des triggers, cette fonction devra être appelée avant tout insertion afin de s'assurer du maintien des DFs. Pensez-vous que dans cette fonction vous auriez du vérifier également que la valeur **codmat** est valide (par rapport à la table *matieres*)? Justifiez votre réponse
.....
.....

Question 4.5. Réalisez autant d'appels que nécessaires pour tester le bon fonctionnement de la fonction en **4.4**.

Normalisation (3 pts)

Question 5.1. Normalisation de la table *notes*.
Si la table *notes* n'est pas en 3NF, proposez une décomposition en deux ou plusieurs tables en 3NF.
Indiquez les noms de tables et les colonnes de chaque table normalisée.
.....
.....
.....
.....
Créez ces nouvelles tables en important les enregistrements existants.

Réaliser uniquement une requête par nouvelle table créée (création et ajout d'enregistrements). A défaut, proposer une solution multi-requêtes.

```
create table notes2 as select note,numet,codmat from notes;
```

– avec perte des matières qui ne sont pas en note

```
create table matieres2 as select nomprof,codmat,coeff from notes natural join matieres group by nomprof, codmat,coeff;
(* puis ajout necessaire des matieres not in notes*)
insert into matieres2 (nomprof,codmat)  select nomprof, codmat from matieres where codmat not in (select codmat from matieres2);
```

– sans perte des matières qui sont pas en note

```
create table matieres3 as (select nomprof,matieres.codmat,coeff from matieres left join notes on matieres.codmat = notes.codmat group by nomprof, matieres.codmat,coeff)
```