Université Henri Poincaré, Nancy 1, Département informatique
Licence informatique-SGBD

TD-TP : Contraintes d'intégrité

1 Cas Centre de Recherche

Le schéma relationnel ci-dessous décrit partiellement l'activité et l'organisation d'un centre de recherche.

Les membres du centre de recherche sont regroupés en équipes de recherche : *Personne.Equ#* désigne le numéro de l'équipe d'appartenance d'une personne. Chaque équipe est dirigée par une personne *(Equipe.Resp#)*. Les équipes conduisent des projets (relation *Developpe*) sur un domaine de recherche (*Projet.Domaine#*). Un projet est sous la direction d'une personne (*Projet.Resp#*) et il peut faire l'objet d'un *contrat*. Plusieurs organismes peuvent contribuer (*Financement.apport*) au financement d'un contrat. Des personnes peuvent participer à plusieurs projets. Les résultats de la recherche peuvent donner lieu à des *publications* dans des revues, journaux et autres congrès. Les publications peuvent avoir plusieurs *auteurs*.

Relations :

Personne(Pers#, Nom, Prenom, Statut, Grade, Equ#)
Equipe(Equ#, Nom_equ, Date_creation, Resp#)
Domaine_Recherche(Domaine#, Designation_En_Clair)
Projet(Nom_proj, Debut, Duree, Resp#, Domaine#)
Contrat(Contr#, Debut, Montant, Nom_proj)
Financement(Contr#, Organisme, Apport)
Developpe(Equ#, Nom_proj)
Participe(Pers#, Nom_proj, Taux_participe)
Publication(Publi#, titre, Date, Domaine#, Nom_proj)
Auteurs(Auteur#, Publi#)

Questions :

1. Décrire les schémas des relations en prenant en compte les contraintes ci-dessous
2. Implanter ces relations
3. Décrire et exécuter un scenario montrant que les contraintes sont bien satisfaites.

Contraintes :

1. Expliciter les dépendances d'inclusion et les clés étrangères. : **cf. clause REFERENCES en faisant attention au références cycliques**.

2. Les noms des équipes de recherche débutent par “LORIA” : **clause CKECK** : (Nom_equ varchar(30) CHECK (Nom_equ LIKE ’LORIA’||’%’)

3. Les publications qui n’ont pas trait à un projet sont rattachées au projet “LORIA-R” : **clause DEFAULT en autorisant les valeurs absentes** (Nom_proj varchar(20) DEFAULT (’LORIA-R’) NULL)

```
DROP TABLE Personne;
CREATE TABLE  Personne (
  Pers#  varchar(13) PRIMARY KEY,
  Nom varchar(30),
  Prenom varchar(30),
  Statut varchar(20)  CHECK (Statut IN
        (’Thesard’, ’McF’, ’CR’, ’DR’, ’Prof’)),
  Grade varchar(5),
  Equ# int NOT NULL /*,
       CONSTRAINT Pers2Equipe
       FOREIGN KEY (Equ#)  REFERENCES Equipe(Equ#)*/);
prompt ’---- Fin CREATION table PERSONNE ’;

DROP TABLE Equipe;
CREATE TABLE Equipe (
      Equ# int PRIMARY KEY,
      Nom_equ varchar(30) CHECK (Nom_equ LIKE ’LORIA’||’%’),
      Date_creation date,
      Resp#  varchar(13),
         CONSTRAINT Equipe2Resp
           FOREIGN KEY (Resp#)
           REFERENCES Personne(Pers#));
prompt ’---- Fin CREATION table Equipe’;

DROP TABLE Domaine_Rech;
CREATE TABLE Domaine_Rech(
     Domaine# varchar(20) PRIMARY KEY,
     Designation_En_Clair varchar(30));

prompt ’---- Fin CREATION table Domaine_Rech’;

DROP TABLE Projet;
CREATE TABLE Projet (
     Nom_proj varchar(20) PRIMARY KEY,
     Debut date,
     Duree date,
     Resp# varchar(13),
     Domaine# varchar(20),
       CONSTRAINT Projet2Resp
```

```
          FOREIGN KEY (Resp#)
          REFERENCES Personne(Pers#),
       CONSTRAINT Projet2Domaine
          FOREIGN KEY (Domaine#)
          REFERENCES Domaine_Rech(Domaine#));
prompt '---- Fin CREATION table Projet';
DROP TABLE Contrat;
CREATE TABLE Contrat (
     Contr# int PRIMARY KEY,
     Debut date,
     Montant numeric(8,2),
     Nom_proj varchar(20),
       CONSTRAINT Contrat2Projet
         FOREIGN KEY (Nom_proj)REFERENCES Projet(Nom_proj));

prompt "---- Fin CREATION table  Contrat";

DROP TABLE Financement;
CREATE TABLE Financement (
     Contr# int,
     Organisme varchar(30),
     Apport     numeric(8,2),
       CONSTRAINT Finance2Contrat
         PRIMARY KEY (Contr#, Organisme),
         FOREIGN KEY (Contr#) REFERENCES Contrat(Contr#));

prompt "---- Fin CREATION table Financement"

DROP TABLE Developpe;
CREATE TABLE Developpe(
    Equ# int,
    Nom_proj varchar(20),
     CONSTRAINT CleDeveloppe
        PRIMARY KEY (Equ#, Nom_proj),
        FOREIGN KEY (Equ#) REFERENCES Equipe(Equ#),
        FOREIGN KEY (Nom_proj)  REFERENCES Projet(Nom_proj));

prompt "---- Fin CREATION table Developpe";

DROP TABLE Participe;
CREATE TABLE Participe (
      Pers# varchar(13),
      Nom_proj varchar(20),
      Taux_participe int NOT NULL
        CHECK (Taux_participe BETWEEN 0 and 100),
```

```
        CONSTRAINT CleParticipe
           PRIMARY KEY (Pers#, Nom_proj),
           FOREIGN KEY (Pers#) REFERENCES Personne(Pers#),
           FOREIGN KEY (Nom_proj)  REFERENCES Projet(Nom_proj));

prompt "---- Fin CREATION table Participe ";

DROP TABLE Publication;
CREATE TABLE Publication (
      Publi# int PRIMARY KEY,
      Titre varchar(30),
      Date_Publi date,
      Domaine# varchar(20),
      Nom_proj varchar(20) DEFAULT ('LORIA') NULL,
         CONSTRAINT Publi2Domaine
           FOREIGN KEY (Domaine#)  REFERENCES Domaine_Rech(Domaine#))

prompt "---- Fin CREATION table Publication ";

DROP TABLE Auteurs;
CREATE TABLE Auteurs(
    Auteur# varchar(13),
    Publi# int,
     CONSTRAINT Auteur2Publi
      PRIMARY KEY (Auteur#,Publi#),
      FOREIGN KEY (Auteur#) REFERENCES Personne(Pers#),
      FOREIGN KEY (Publi#)  REFERENCES Publication (Publi#));

prompt "---- Fin CREATION table Auteurs (derniere)"
```

4. La somme des taux de participation (*Participe.Taux_participe*) d'une personne à des projets ne peut pas être négative comme elle ne peut pas excéder 100% : **2 triggers "after insert or update of Taux_Participe", le premier *"for each row"* et le second sur instruction (pas *"for each row"*)** :

   (a) Créer une table temporaire pour y enregistrer les modifications. Ne pas oublier la clause *on commit preserve rows* pour éviter que les lignes insérées ne soient détruites à la fin du trigger :
   create temporary table Temp2(pers#, taux_participe) on commit preserve rows;

   (b) Créer le premier trigger qui enregistre les modifications dans la table temporaire :

```
create or replace trigger Participe_1
after insert or update of Taux_Participe on Participe
for each row
```

```
when (new.taux_participe > old.taux_participe)
begin
  dbms_output.put_line ('-- Trigger Participe-1');
  insert into Temp2(x, y)
         values (:new.pers#, :new.taux_participe);
end;
/
```

(c) Créer le second trigger qui vérifie que la contrainte n'est pas violée :

```
create or replace trigger Participation_3
after  insert or update of Taux_Participe on Participe
declare
  Somme_Taux int;
  Nbr_Sup_100 int;
begin
  dbms_output.put_line ('-- Trigger Participe-2');
  select count(*) into Nbr_Sup_100 from Temp2;
  dbms_output.put_line
   ('-- Trigger Participe-2: Card(Temp2) = '||Nbr_Sup_100);
  select count(*) into Nbr_Sup_100
  from Participe p, Temp2 t where p.pers# = t.x
  group by p.pers# having (sum(Taux_participe) > 100);
  dbms_output.put_line
     ('-- Trigger Participe-2: Nbr_Sup_100 = '||Nbr_Sup_100);
  if Nbr_Sup_100 > 0
  then raise_application_error(-20002,
         'Somme des taux de participation > 100 !');
  end if;
end;
/
```

5. Le statut des personnes appartient à l'ensemble de valeurs {Thesard, McF, Prof, CR, DR}, celles-ci signifiant, respectivement thésard, maître de conférences, professeur, chargé de recherche et directeur de recherche. Le changement de statut doit se conformer au diagramme de transition décrit par la table de la figure 1 où $T(i, j) = O$ signifie que la transition de i à j est autorisée : **un trigger sur mise à jour de Personne.Statut** :

```
create or replace trigger TransitStatut2
before update of Statut on Personne
for each row
begin
  if (:old.Statut  in ('Thesard')
       and :new.Statut NOT in ('Thesard',  'McF', 'CR')
```

```
        )
       or ( :old.Statut in ('McF')
            and :new.Statut NOT in ('McF',  'CR', 'DR',  'Prof')
          )
       or ( :old.Statut  in ('CR')
             and :new.Statut NOT in ('McF',  'CR', 'DR',  'Prof')
          )
       or ( :old.Statut  in ('DR')
             and :new.Statut NOT in ('DR',  'Prof')
           )
        or ( :old.Statut in ('Prof')
                   and :new.Statut  NOT in ('DR',  'Prof')
           )
    then begin
     raise_application_error(-20001,'Changement de statut  incorrect');
        end;
    end if;
end;
```

↗	Thesard	McF	CR	DR	Prof
Thesard	O	O	O	N	N
McF	N	O	O	O	O
CR	N	O	O	O	O
DR	N	N	N	O	O
Prof	N	N	N	O	O

Figure 1: Table de transition de l'attribut Personne.Statut

Questions :

1. Décrire les schémas des relations et leurs contraintes.
2. Implanter ces relations
3. Décrire et exécuter un scenario montrant que les contraintes sont bien satisfaites.

2 Cas gestion de stocks

Soient les relations :

- produit (prod#,libelle,pu)
- stock (prod#,dep#,qte)
- depot (dep#,adr,capacite)

et les contraintes portant sur ces relations :

1. pu est obligatoire dans la relation produit : **CLAUSE CHECK** ;
2. tous les attributs de la relation stock sont obligatoires : **CLAUSE NOT NULL** ;
3. la capacité minimum d'un dépôt est de 1000 : **CLAUSE CHECK** ;
4. les numéros de produits sont inférieurs à 10000 : **CLAUSE CHECK** ;
5. un produit disparaît de la base dès qu'il n'est plus stocké dans aucun dépôt : **cf. la solution pour Taux_Partcipe de l'exercice précédent** ;
6. un produit ne peut pas être stocké dans plus de trois dépôts ;
7. les produits 3 et 5 ne doivent pas être stockés dans le même dépôt ;
8. les produits 4, 8, 12 ne sont stockés que dans le dépôt 3.

3 Notion de données dérivées

Ajouter une colonne *Total_Stock* à la table produit (cf. *alter table . . . add column*). Cette colonne est destinée à contenir la quantité totale en stock d'un produit. Ecrire, mettre au point et tester un trigger qui permet de produire et maintenir sa valeur :

1. Modifier le schéma de la relation : *alter table produit add(Total_Stock int default 0)*;
2. Créer un trigger *insert or update of qte on stock* qui distingue la sortie et l'entrée en stock.

```
/* Ajout de la colonne Total_Stock  à produit */
SQL> alter table produit add (Total_Stock int) int default 0;

/* INITIALISATION de la colonne ajoutée        */
SQL> update produit p set p.total_stock =
  2  (select sum(s.qte) from stock s
  3  where s.prod# = p.prod#
  4  group by s.prod#);

SQL> select * from stock where prod# = 1;

     PROD#       DEP#        QTE
---------- ---------- ----------
         1          2        202
         1          3          3
         1          1          1

SQL> select total_stock from produit where prod# = 1;
```

```
TOTAL_STOCK
-----------
        206

/* Un trigger réalisant entrée et sortie de stock */
/* + un trigger pour le cas de la 1ère insertion  */
/* + un trigger en cas de suppression             */
/* SOLUTION "OPTMISABLE", je pense. mais je       */
/* n'ai pas le temps....                          */
/*   Pour l'affichage de la trace d'exécution     */
/* NE PAS OUBLIER : SQL> set serveroutput on      */

/*--- INSERTION dans STOCK          ---*/
create or replace trigger Derived_Data_INSERT
after insert on stock for each row
begin
       dbms_output.put_line('Derived_Data_INSERT: Trigger
                                  Insertion dans stock');
       update nacerb.produit p set
         Total_Stock = Total_Stock + :new.qte
       where p.prod# = :new.prod#;
end;
/*--- UPDATE dans STOCK          ---*/
create or replace trigger Derived_Data_UPDATE
after update of qte on stock for each row
begin
dbms_output.put_line('Entrée du trigger Derived_Data_UPDATE');
  if :new.qte < :old.qte
  then
       dbms_output.put_line('Sortie de stock');
       update nacerb.produit p set
         Total_Stock = Total_stock -(:old.qte - :new.qte)
       where p.prod# = :new.prod#;
  else
       dbms_output.put_line('Entree de stock');
       update nacerb.produit p set
         Total_Stock = Total_stock + (:new.qte - :old.qte)
       where p.prod# = :new.prod#;
  end if;
  dbms_output.put_line('Sortie du trigger Derived_Data_UPDATE');
end;
/*--- DELETE dans STOCK          ---*/
create or replace trigger Derived_Data_DELETE
after delete on stock for each row
begin
```

```
        dbms_output.put_line('Trigger Derived_Data_DELETE:
                                    Suppression dans stock');
        update nacerb.produit p set
          Total_Stock = Total_Stock - :old.qte
        where p.prod# = :old.prod#;
end;
```

4 Gestion des droits

Se mettre par groupe de trois : soit U1, U2 et U3 les membres d'un groupe. Exécuter la séquence ci-dessous. Justifier le résultat.

1. U1 donne à U2 le droit d'interroger (droit select) sa table produit ;
2. U2 crée une vue sur la table U1.produit ;
3. U2 donne à U3 le droit d'interroger sa vue ;
4. U3 interroge la vue de U2.

Corrigé : U3 n'a pasle droit d'interroger la vue puisque la vue est construite sur U1.produit et que U3 n'a pas le droit d'interroger U1.produit.

5 Dictionnaire Oracle

Ecrire et exécuter les requêtes SQL répondant aux questions ci-dessous :

1. Quelles sont les colonnes du dictionnaire Oracle (DICTIONARY ou DICT) ?

```
 SQL> desc dict
Nom                    NULL ?   Type
---------------------- -------- ----------------
TABLE_NAME                      VARCHAR2(30)
COMMENTS                        VARCHAR2(4000)
```

2. Quel est le nombre de tuples de cette table ?

```
SQL> select count(*) from dict;
  COUNT(*)
----------
      1362
```

3. Quelles sont les tables et les vues du dictionnaire qui ont trait aux LOGs?
   Note : Compte tenu de la réponse à la question précédente, il est préférable de faire une recherche avec *motif*, comme montré en cours, i.e. :

select table_name from dict
where upper(table_name) like upper('%LOG%');

4. Quelles sont celles qui ont trait aux espaces de stockage (TABLESPACE) ?

```
SQL> select table_name from dict
  2  where upper(table_name) like upper('%tablespace%');

TABLE_NAME
------------------------------
USER_TABLESPACES
...
V$TABLESPACE
GV$TABLESPACE
```

5. Quelles sont les tablespaces existantes ?

```
Après un DESC user_tablespaces, choisir la "bonne" colonne.
SQL> select TABLESPACE_NAME from user_tablespaces;

TABLESPACE_NAME
------------------------------
LMD_IDX
LMD
```

6. Quelles sont les tables et les vues du dictionnaire qui ont trait aux espaces de données (DATAFILE) ?
   cf. ce qui précède : select table_name from dict ... like ....

7. Donner, pour chaque datafile, son nom ainsi que le nom de la tablespace à laquelle il appartient.
   Jointure entre V$TABLESPACE et V$DATAFILE.

8. Quels sont les noms des tables que vous possédez (USER_TABLES) ?

9. Quel est le nombre de tables auxquelles vous avez accès (ALL_TABLES) ?

10. Quelles sont les tables qui ont trait aux triggers ?

11. Donne le nom et l'état (*status*) de chacun de vos triggers.

12. Quelles sont les actions (ou privilèges) que vous avez le droit d'exécuter pendant la session (*sys.session_privs*) ?

```
SQL> select PRIVILEGE from session_privs;

PRIVILEGE
```

```
----------------------------------------
CREATE SESSION
CREATE TABLE
CREATE PROCEDURE
CREATE TRIGGER
```