SGBD : BASES DE DONNÉES AVANCÉES [M3106C]

TD $N^0$6 - CONTRAINTES D'INTÉGRITÉ

Objectifs

– Mise en oeuvre des contraintes
– Triggers

Corrigés

Exercice I :

Question 1.1.
```
UPDATE  etudiant SET etudiant_id=
    CASE WHEN ((SELECT res_note FROM resultat
                  WHERE res_etudiant=etudiant_id
                  AND res_ue='1') >=10)
    THEN '1' ELSE
    CASE WHEN ((SELECT res_note FROM resultat
                  WHERE res_etudiant=etudiant_id
                  AND res_ue='1') < 8.00)
    THEN '3' ELSE '2' END END  ||etudiant_id;
```

Question 1.2.
```
#   BEGIN TRANSACTION;
BEGIN

#   UPDATE  etudiant SET etudiant_id=
      CASE WHEN ((SELECT res_note FROM resultat
                    WHERE res_etudiant=etudiant_id
                    AND res_ue='1') >=10)
      THEN '1' ELSE
      CASE WHEN ((SELECT res_note FROM resultat
                    WHERE res_etudiant=etudiant_id
                    AND res_ue='1') < 8.00)
      THEN '3' ELSE '2' END END  ||etudiant_id;
UPDATE 10
```

Date: 30 septembre 2014.
Hocine ABIR - IUT Villetaneuse .

```
#   SELECT * FROM Etudiant;
 etudiant_id | etu_nom_prenom
-------------+-----------------
 1001        | LEBEUF-Martin
 2002        | MARTINEZ-Dupont
 1003        | CARLIN-Dubois
 1004        | RIDLEY-Durant
 3005        | CONEN-Dupont
 3006        | INDESIT-Jean
 1007        | LEE-Didier
 1008        | MALONGA-Pierre
 1009        | LEDIS-Alex
 2010        | MARC-Olivier
(10 rows)

#   ROLLBACK;
ROLLBACK
```

Question 1.3.
```
#  ALTER TABLE resultat
    ADD CONSTRAINT resultat_fk
    FOREIGN KEY (res_etudiant)
    REFERENCES   etudiant(etudiant_id);
ALTER TABLE
```

Question 1.4.
```
# select c.conname as "CI",t.relname as "ON",c.contype as "Ty
     FROM pg_constraint c, pg_class t
     where c.conrelid=t.oid
     and t.relname in ('resultat','etudiant');
          CI          |    ON    | Type
----------------------+----------+------
 etudiant_pk          | etudiant | p
 inavlide_resultat_ue | resultat | c
 resultat_pk          | resultat | p
 resultat_fk          | resultat | f
(4 rows)

# select c.conname as "CI",t.relname as "ON",c.condeferred as "Diff",
   r.relname as "Reference" ,confupdtype as "UPD",confdeltype as "DEL"
     FROM pg_constraint c, pg_class t ,pg_class r
     where c.contype='f'
     and c.conrelid=t.oid
     and c.confrelid=r.oid
     and t.relname='resultat';
     CI      |    ON    | Diff | Reference | UPD | DEL
-------------+----------+------+-----------+-----+-----
 resultat_fk | resultat | f    | etudiant  | a   | a
(1 row)

# select tgconstrname as "CI",p.proname as "Funct Name (Trigger)",
    t.relname as "ON",r.relname as "Reference"
     from pg_class t, pg_trigger g,pg_class r,pg_proc p
     where tgrelid=t.oid
     and t.relname in   ('resultat','etudiant')
     and tgconstrrelid=r.oid
     and p.oid=g.tgfoid;
     CI      | Funct Name (Trigger) |    ON    | Reference
-------------+----------------------+----------+-----------
 resultat_fk | RI_FKey_check_ins    | resultat | etudiant
 resultat_fk | RI_FKey_check_upd    | resultat | etudiant
 resultat_fk | RI_FKey_noaction_del | etudiant | resultat
 resultat_fk | RI_FKey_noaction_upd | etudiant | resultat
(4 rows)
```

Question 1.5. *La requête de la question (**??**) échoue (abort) à cause de la contrainte d'intégrité référencielle* `resultat_fk`*.*

```
  update  etudiant set etudiant_id=case when
   ((select res_note from resultat
       where res_etudiant=etudiant_id
       and res_ue='1') >=10) then '1' else
  case when ( (select res_note from resultat
               where res_etudiant=etudiant_id
               and res_ue='1')<8.00)
  then '3' else '2' end end  ||etudiant_id;

 ERROR:  update or delete on "etudiant" violates \
   foreign key constraint "resultat_fk" on "resultat"
DETAIL:  Key (etudiant_id)=(001 ) is \
        still referenced from table "resultat".
```

Question 1.6. `# begin transaction;`

```
BEGIN

# alter table resultat
```

```
   drop constraint resultat_fk;
ALTER TABLE

#  alter table resultat
    add constraint resultat_fk
    foreign key(res_etudiant)
        references etudiant(etudiant_id)
       on update cascade;
ALTER TABLE

 -- mise a jour

#  update  etudiant set etudiant_id=case when
    ((select res_note from resultat
        where res_etudiant=etudiant_id
        and res_ue='1') >=10) then '1' else
   case when ( (select res_note from resultat
                where res_etudiant=etudiant_id
                 and res_ue='1')<8.00)
    then '3' else '2' end end  ||etudiant_id;
UPDATE 10

    -- visualisation

#   select * from etudiant;
 etudiant_id | etu_nom_prenom
-------------+-----------------
 1001        | LEBEUF-Martin
 2002        | MARTINEZ-Dupont
 1003        | CARLIN-Dubois
 1004        | RIDLEY-Durant
 3005        | CONEN-Dupont
 3006        | INDESIT-Jean
 1007        | LEE-Didier
 1008        | MALONGA-Pierre
 1009        | LEDIS-Alex
 2010        | MARC-Olivier
(10 rows)

#   select * from resultat;
 res_etudiant | res_ue | res_note
--------------+--------+----------
 1001         | 1      |    14.50
 1001         | 2      |    12.75
 2002         | 1      |     9.25
 1003         | 1      |    10.00
 1003         | 2      |    10.00
 1003         | 3      |     6.50
 1004         | 1      |    12.00
 3005         | 1      |     4.50
 3006         | 1      |     7.50
 1007         | 1      |    13.50
 1008         | 1      |    16.50
 1009         | 1      |    11.50
 2010         | 1      |     8.00
(13 rows)

# rollback;
ROLLBACK
```

Question 1.7. `# begin transaction;`

```
BEGIN

# alter table resultat
    drop constraint resultat_fk;
ALTER TABLE

# alter table resultat
    add constraint resultat_fk
        foreign key(res_etudiant)
        references etudiant(etudiant_id)
        INITIALLY IMMEDIATE DEFERRABLE;
ALTER TABLE

#    SET CONSTRAINTS resultat_fk DEFERRED ;
SET CONSTRAINTS

      -- maj table Etudiant

#   update  etudiant set etudiant_id=case when
      ((select res_note from resultat
          where res_etudiant=etudiant_id
          and res_ue='1') >=10) then '1' else
     case when ( (select res_note from resultat
                  where res_etudiant=etudiant_id
                  and res_ue='1')<8.00)
     then '3' else '2' end end  ||etudiant_id;
```

```
UPDATE 10

    -- maj table resultat

#     update  resultat  set res_etudiant =
      (select etudiant_id from etudiant
         where substr(etudiant_id,2,length(etudiant_id)-1) =
         substr(res_etudiant,1,length(res_etudiant)) );
UPDATE 13

# commit;
COMMIT

# select * from etudiant;
 etudiant_id | etu_nom_prenom
-------------+-----------------
 1001        | LEBEUF-Martin
 2002        | MARTINEZ-Dupont
 1003        | CARLIN-Dubois
 1004        | RIDLEY-Durant
 3005        | CONEN-Dupont
 3006        | INDESIT-Jean
 1007        | LEE-Didier
 1008        | MALONGA-Pierre
 1009        | LEDIS-Alex
 2010        | MARC-Olivier
(10 rows)

#   select * from resultat;
 res_etudiant | res_ue | res_note
--------------+--------+----------
 1001         | 1      |    14.50
 1001         | 2      |    12.75
 2002         | 1      |     9.25
 1003         | 1      |    10.00
 1003         | 2      |    10.00
 1003         | 3      |     6.50
 1004         | 1      |    12.00
 3005         | 1      |     4.50
 3006         | 1      |     7.50
 1007         | 1      |    13.50
 1008         | 1      |    16.50
 1009         | 1      |    11.50
 2010         | 1      |     8.00
```

```
(13 rows)
```

Question 1.8.

(1) *INSERT Resultat*

```
CREATE FUNCTION  check_ins () RETURNS TRIGGER AS '
    DECLARE
        tuple record;
    BEGIN
        IF NEW.res_etudiant ISNULL THEN
           RETURN NEW;
        END IF;
        SELECT INTO tuple * FROM etudiant
            WHERE etudiant_id = new.res_etudiant;
        IF NOT FOUND THEN
             RETURN NULL;
        END IF;
        RETURN NEW;
    END;
' LANGUAGE 'plpgsql';
CREATE TRIGGER ins_no_resultat
    BEFORE INSERT ON resultat
    FOR EACH ROW EXECUTE PROCEDURE check_ins();
```

(2) *UPDATE Resultat*

```
CREATE FUNCTION  check_upd () RETURNS TRIGGER AS '
    DECLARE
        tuple record;
    BEGIN
        IF NEW.res_etudiant ISNULL THEN
           RETURN NEW;
        END IF;
        SELECT INTO tuple * FROM etudiant
            WHERE etudiant_id = new.res_etudiant;
        IF NOT FOUND THEN
             RETURN NULL;
        END IF;
        RETURN NEW;
    END;
' LANGUAGE 'plpgsql';
CREATE TRIGGER upd_no_resultat
    BEFORE UPDATE ON resultat
    FOR EACH ROW EXECUTE PROCEDURE check_upd();
```

(3) *DELETE Etudiant*

```
CREATE FUNCTION  no_action_del () RETURNS TRIGGER AS '
    DECLARE
        tuple record;
    BEGIN
        SELECT INTO tuple * FROM resultat
            WHERE res_etudiant = old.etudiant_id;
        IF FOUND THEN
             RETURN NULL;
        END IF;
        RETURN OLD;
    END;
' LANGUAGE 'plpgsql';
CREATE TRIGGER del_no_etudiant
    BEFORE delete ON etudiant
    FOR EACH ROW EXECUTE PROCEDURE no_action_del();
```

(4) *UPDATE Etudiant*

```
CREATE or REPlace FUNCTION  no_action_upd () RETURNS TRIGGER AS '
    DECLARE
        tuple record;
    BEGIN
        IF NEW.etudiant_id = old.etudiant_id THEN
           RETURN NEW;
        END IF;
        SELECT INTO tuple * FROM resultat
            WHERE res_etudiant = old.etudiant_id;
        IF FOUND THEN
             RETURN NULL;
        END IF;
        RETURN NEW;
    END;
' LANGUAGE 'plpgsql';
CREATE TRIGGER upd_no_etudiant
    BEFORE UPDATE ON etudiant
    FOR EACH ROW EXECUTE PROCEDURE no_action_upd();
```

Question 1.9. *Créer un index secondaire sur la table* `resultat` *ayant pour clé la clé, la clé étrangère* `res_etudiant`