`06/10/2017 a 11h`
`Objet du mail: [AdminBD] [TP1]`

Compte Rendue AdminBD -- TD1

Yuxin SHI

Création et contraintes

1. Créer un utilisateur

```
-- create user
CREATE USER tearsyu IDENTIFIED BY toto;
```

2. Créer un tablespace de 30 Mo.

```
-- create tablespaces of 30Mo
CREATE TABLESPACE tb_td1 DATAFILE 'tb_td1.dbf' SIZE 30M ONLINE;
```

3. Donner les autorisations nécessaires à l'utilisateur :
   - connexion
   - quota de 10 Mo sur le tablespace créé à la question 2.
   - création de tables.
   - création de déclencheurs.

```
-- Permission controller of uses
GRANT CONNECT TO tearsyu; --tearsyu is able to connect to oracle
Alter USER tearsyu quota 10M on tb_td1; --tearsyu has 10M to use on tablespac
Alter USER tearsyu default TABLESPACE tb_td1; --tearsyu will connect to it's
GRANT CREATE TABLE TO tearsyu;
GRANT CREATE TRIGGER TO tearsyu;
```

4. Créer les tables du TD précédent en spécifiant les clés primaires et étrangères.

```
--Create table
CREATE TABLE responsable(
   nr               int,
   nom               varchar(50) not null,
   prenom             varchar(50) not null,
   dpt               varchar(50) not null,
   primary key (nr)
```

```
);

CREATE TABLE cours(
   nc               int primary key,
   code_cours     varchar(50) not null,
   intitule       varchar(50) not null,
   ects             varchar(50) not null,
   nr               int not null,
 dpt          varchar(50) not null,
 foreign key (nr) references responsable(nr)
);

CREATE TABLE etudiant(
   ne              int,
   nom              varchar(50) not null,
   prenom           varchar(50) not null,
   ville            varchar(50) not null,
 age         int,
   primary key (ne)
);

CREATE TABLE inscrit(
 ne          int not null,
 nc          int not null,
 annee       int not null,
 primary key(ne, nc, annee),
 foreign key (ne) references etudiant(ne),
 foreign key (nc) references cours(nc)
);

CREATE TABLE resultat(
 ne          int not null,
 nc          int not null,
 annee       int not null,
 note        int not null,
 foreign key (ne, nc, annee) references inscrit(ne, nc, annee)
);
```

5. Donner des ordres SQL permettant de vérifier que les clés primaires et étrangères

```
--Insert data
INSERT INTO responsable VALUES (1, 'Michael', 'Mico', 'informatique');
INSERT INTO responsable VALUES (2, 'Demail', 'Logic', 'informatique');
INSERT INTO responsable VALUES (3, 'Yuxin', 'SHI', 'securite');

INSERT INTO etudiant VALUES(1, 'Zixi', 'DENG', 'Melborne', 26);
INSERT INTO etudiant VALUES(2, 'Anais', 'Anais','Paris', 23);
INSERT INTO etudiant VALUES(3, 'Zadi', 'Laetitia', 'Paris', 22);

INSERT INTO cours VALUES(1, 'INFO_01', 'BD', 4, 1, 'informatique');
INSERT INTO cours VALUES(2, 'INFO_02', 'Prog', 5, 3, 'informatique');
INSERT INTO cours VALUES(3, 'INFO_03', 'Network', 3, 3, 'securite');
INSERT INTO cours VALUES(4, 'ARCHI_01', 'Design', 4, 2, 'architecture');
```

```
INSERT INTO inscrit VALUES (1, 1, 2015);
INSERT INTO inscrit VALUES (1, 2, 2016);
INSERT INTO inscrit VALUES (1, 2, 2017);
INSERT INTO inscrit VALUES (2, 3, 2015);
INSERT INTO inscrit VALUES (3, 1, 2015);
```

Ici si on insère une colonne avec les clés primares ou les clés étrangères qui n'existent pas, par ex:

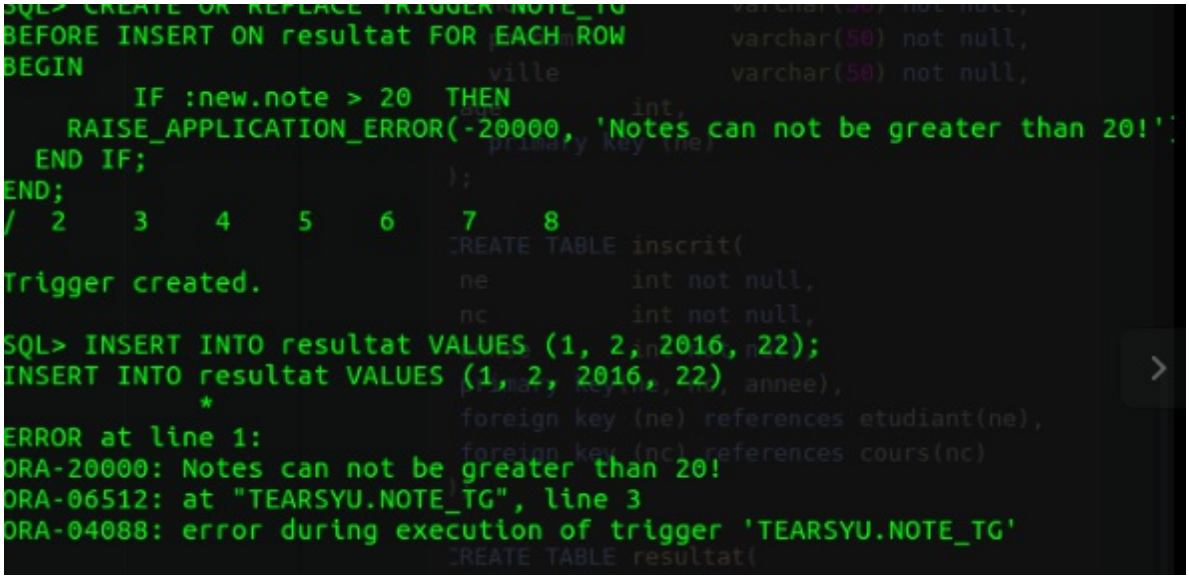
```
INSERT INTO responsable VALUES(2, 'Dupont', 'Sachat');
INSERT INTO inscrit VALUES(5, 2, 2011);
```

Il y aura des erreurs.

1. Définir les deux déclencheurs demandés dans la question 4 du TD précédent.
2. Donner des ordres SQL permettant de vérifier que ces deux déclencheurs sont bien mis en œuvre.

```
-- Create trigger 1
CREATE OR REPLACE TRIGGER NOTE_TG
BEFORE INSERT ON resultat FOR EACH ROW
BEGIN
   IF :new.note > 20  THEN
   RAISE_APPLICATION_ERROR( -20000, 'Notes can not be greater than 20!');
 END IF;
END;
/
```

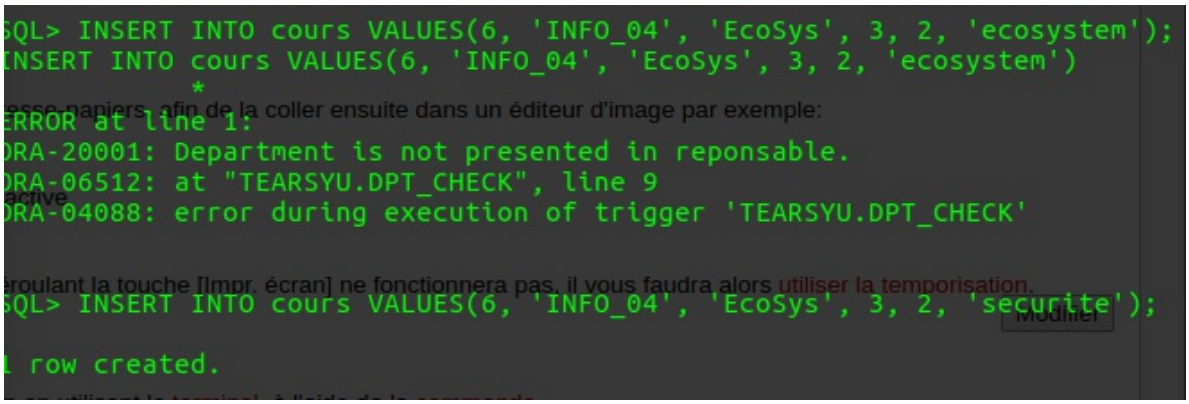
After creating this trigger, we can test if trigger 1 works:



There we see it's impossible to insert a row at cours table if the note is greater than 20, or the error is raised.

```
CREATE OR REPLACE TRIGGER DPT_CHECK
BEFORE INSERT ON cours FOR EACH ROW
DECLARE
dptr varchar(50);
newdpt varchar(50);
BEGIN
SELECT :new.dpt INTO newdpt FROM dual;
SELECT responsable.dpt into dptr FROM responsable
WHERE newdpt = responsable.dpt;
EXCEPTION WHEN NO_DATA_FOUND TH
 RAISE_APPLICATION_ERROR( -20001, 'Department is not presented in reponsable.');
END;
/
```

This screenshot shows how this trigger works:

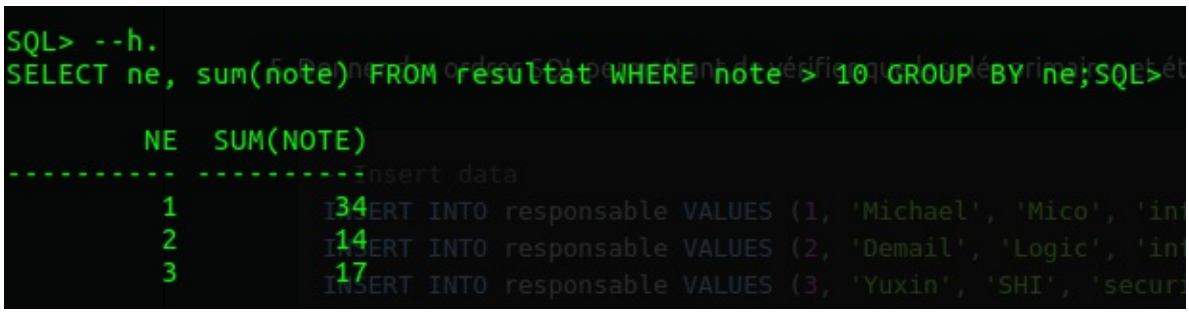


There if I insert a row with dpt "ecosystem" which doesn't exist in responsable, the error that I create at the trigger is raised.

3. Donner les requêtes h et i du TD précédent.

```
--h.
SELECT ne, sum(note) FROM resultat WHERE note > 10 GROUP BY ne;
--i
SELECT count(etudiant.ne), max(etudiant.age), avg(resultat.note) as moyenne
FROM etudiant INNER JOIN resultat ON resultat.ne = etudiant.ne
INNER JOIN cours ON resultat.nc = cours.nc
WHERE cours.dpt='informatique'
AND (SELECT avg(note) FROM resultat) >=12 GROUP BY cours.dpt;
```

resultat de h:



resultat de i:

```
SQL> SELECT count(etudiant.ne), max(etudiant.age), avg(resultat.note) as moyenne
FROM etudiant INNER JOIN resultat ON resultat.ne = etudiant.ne
INNER JOIN cours ON resultat.nc = cours.nc
WHERE cours.dpt='informatique' AND (SELECT avg(note) FROM resultat) >=12
GROUP BY cours.dpt;  2    3    4    5

COUNT(ETUDIANT.NE) MAX(ETUDIANT.AGE)    MOYENNE
------------------ ------------------ ----------
                 4                 26      15.25
```

Interrogation du dictionnaire

1. Donner la liste des tables créées précédemment.

```
--Use tearsyu
SELECT table_name FROM user_tables;
```

```
SQL> SELECT table_name FROM user_tables;

TABLE_NAME
------------------------------
RESPONSABLE
COURS
ETUDIANT
INSCRIT
RESULTAT
```

2. Donner la liste les contraintes créées précédemment.

```
--Use tearsyu
SELECT * FROM USER_CONSTRAINTS;
```

```
CONSTRAINT_NAME                C TABLE_NAME
------------------------------ - ------------------------------
SEARCH_CONDITION
--------------------------------------------------------------------------------
R_OWNER
------------------------------
R_CONSTRAINT_NAME              DELETE_RU STATUS   DEFERRABLE     DEFERRED
------------------------------ --------- -------- -------------- ---------
VALIDATED      GENERATED       BAD RELY LAST_CHANGE
-------------- --------------- --- ---- -----------
INDEX_OWNER                    INDEX_NAME                     INVALID
------------------------------ ------------------------------ -------
VIEW_RELATED
--------------


28 rows selected.
```

3. Donner la liste des tablespace.

```sql
--Use sys
SELECT TABLESPACE_NAME, STATUS FROM USER_TABLESPACES;
```

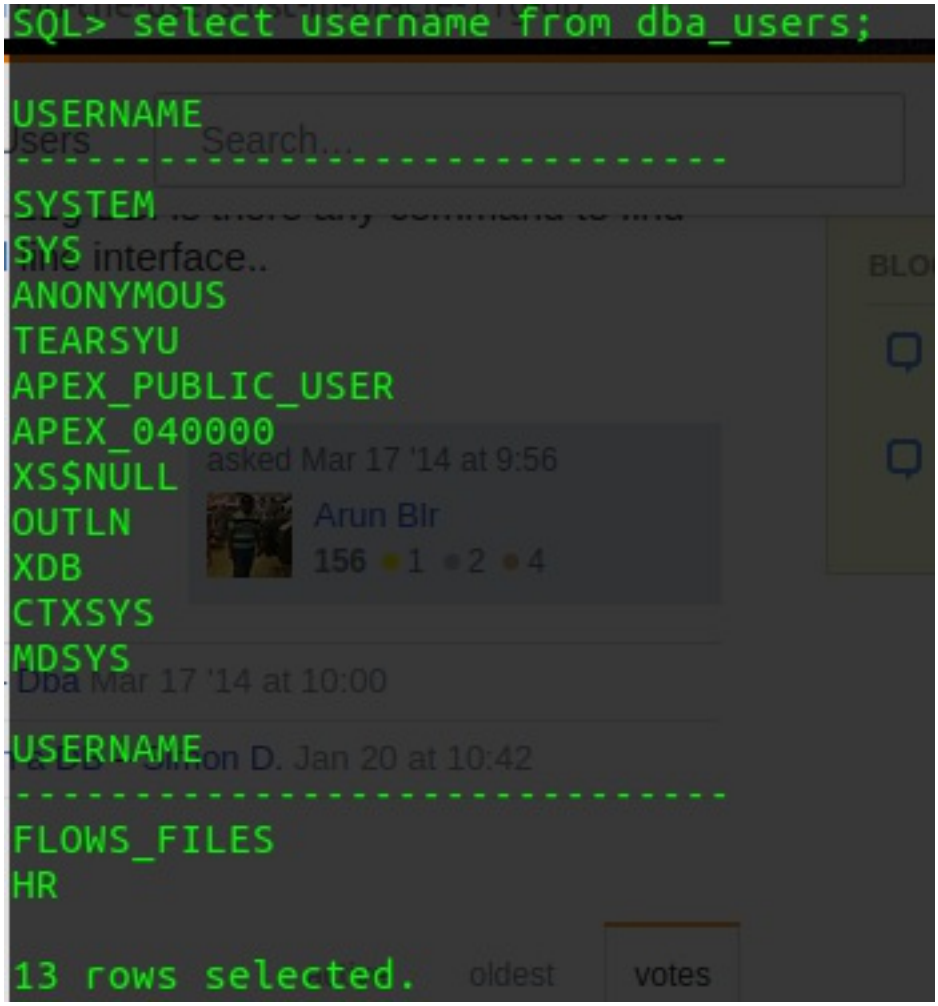
```
SQL> select tablespace_name, status from user_tablespaces;

TABLESPACE_NAME                STATUS
------------------------------ ---------
SYSTEM                         ONLINE
SYSAUX                         ONLINE
UNDOTBS1                       ONLINE
TEMP                           ONLINE
USERS                          ONLINE
TB_TD1                         ONLINE

6 rows selected.
```

4. Donner la liste des utilisateurs.

```sql
--Use sys
SELECT username FROM dba_users;
```


SQL> select username from dba_users;
USERNAME
SYSTEM
SYS
ANONYMOUS
TEARSYU
APEX_PUBLIC_USER
APEX_040000
XS$NULL
OUTLN
XDB
CTXSYS
MDSYS
USERNAME
FLOWS_FILES
HR
13 rows selected.