RAPPORT DE PROJET JEE

Partie PL/SQL

Diallo Thierno Yaya
2016 – 2017
Application Informatique (JEE)
Brigitte Copin

Sujet :

Nous avons décidé de mettre en place une plateforme de gestion des stages. Le but principal est de permettre aux différents acteurs (Etudiants, Superviseurs, coordinatrice) impliquer dans le processus de stage de pouvoir consulter les informations concernant le stage d'un élève donné.

Cela permettra aussi de savoir précisément qui contacter en cas de problème et à qui envoyer les différents documents. Le programme sera principalement alimenté par la personne qui s'occupe des conventions de stage, ainsi que l'étudiant qui pourra entre ses informations directement sur la plateforme et choisir son professeur référent.

La base de données

Schéma MCD :

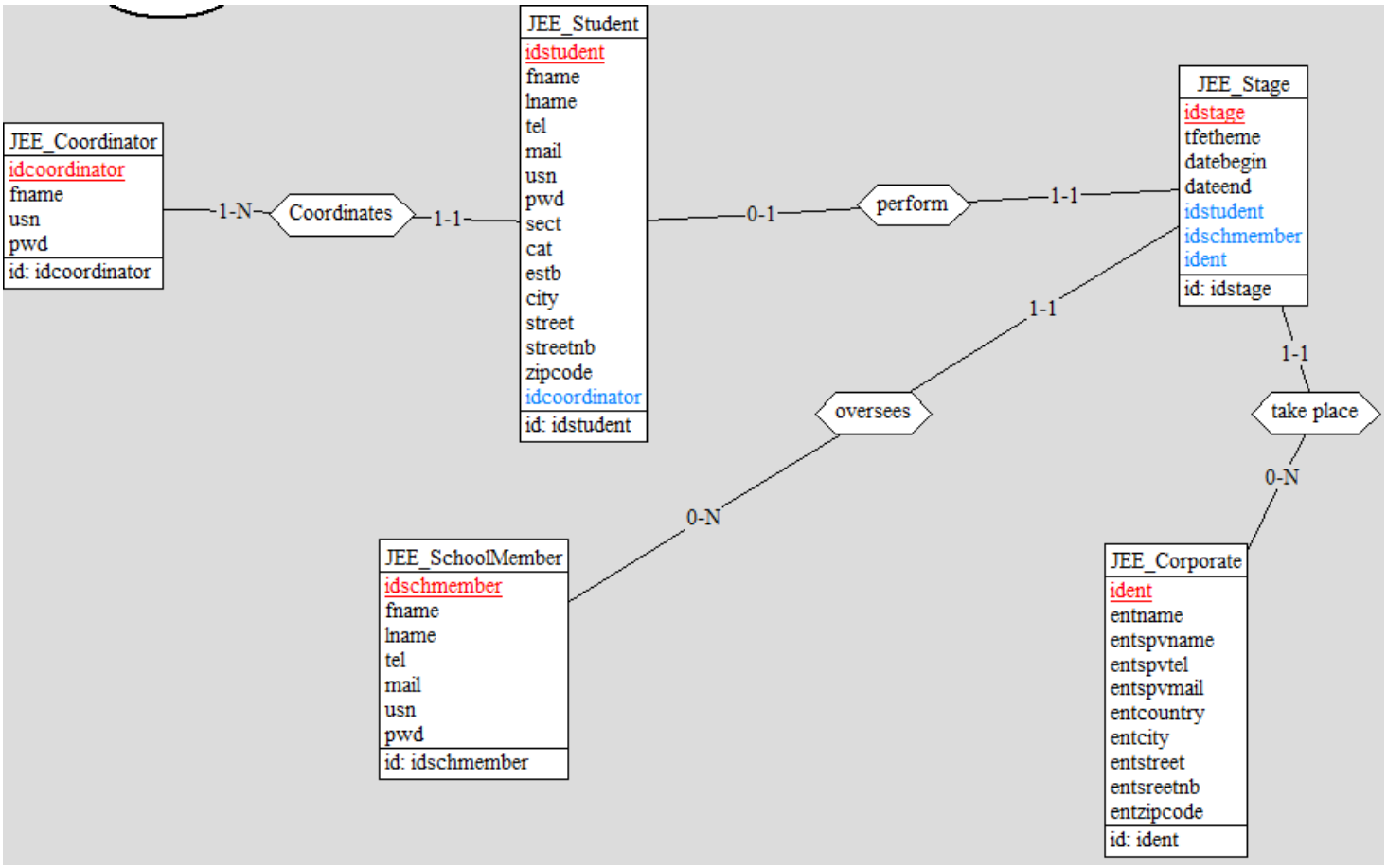

FIGURE 1: SCHEMA DE LA BASE DE DONNEES

Explication

Une coordinatrice (JEE_Coordinator) possède un identifiant (idcoordinator), un login (usn) et un mot de passe (pwd) ainsi qu'une information classique (NAME).

Elle peut coordonner un ou plusieurs Etudiants (JEE_STUDENT) qui possèdent des informations classiques telles que : NAME, MAIL, LOGIN, MOT DE PASSE, ADDRESSE ainsi que les informations relatives à l'établissement d'enseignement.

Chaque Etudiant effectue un Stage (JEE_STAGE) qui contient des informations sur son Thème du TFE(TFETHEME), la date de début et de fin. Cette dernière table est reliée à deux autres tables qui sont respectivement JEE_CORPORATE et JEE_SCHOOLMEMBER.

La table JEE_CORPORATE renferme les informations sur l'entreprise de stage Nom Entreprise (entname), l'adresse de l'entreprise et des informations sur le superviseur du stage (nom, email et téléphone).

La dernière table JEE_SCHOOLMEMBER contient des informations sur le professeur chargé de superviser le stage de l'étudiant. Il est composé des informations classique (nom, prénom, email, tél, login et mot de passe).

Script de création de la base de données

```
/*==============================================================*/
/*          PROJET JEE vs PL/SQL                                */
/*NOM  SGBD        :  ORACLE Version 11g                        */
/*NOM DE LA DB     :  Gestion de stage                          */
/*DATE DE CREATION :  12/07/2017 09:00:00                       */
/*==============================================================*/

/*DROP USER Yaya_JEE CASCADE;
CREATE USER Yaya_JEE IDENTIFIED BY rama;
GRANT CONNECT, RESOURCE, DBA to STAGE;

CONNECT Yaya_JEE/rama*/


/*==============================================================*/
/* Table : JEE_Coordinator                                      */
/*==============================================================*/
create table JEE_Coordinator (
   idcoordinator      INTEGER       not null,
   fname      VARCHAR2(100),
   usn        VARCHAR2(100)     not null,
   pwd        VARCHAR2(100) not null,
   constraint PK_JEE_Coordinator primary key (idcoordinator)
);

/*==============================================================*/
/* Table : JEE_Student                                          */
/*==============================================================*/
create table JEE_Student (
   idstudent      INTEGER   not null,
   fname    VARCHAR2(100)   not null,
   lname    VARCHAR2(100)   not null,
   tel      VARCHAR2(50)    not null,
   mail     VARCHAR2(100)   not null,
   usn      VARCHAR2(100)   not null,
   pwd      VARCHAR2(100)   not null,
   sect     VARCHAR2(50),
   cat      varchar2(50),
   estb     varchar2(100),
   city     VARCHAR2(100),
   street   VARCHAR2(60),
   streetnb INTEGER,
   zipcode  INTEGER,
   idcoordinator      INTEGER not null,
   constraint PK_JEE_Student primary key (idstudent)
);
```

```
/*==============================================================*/
/* Index : Coordinates_FK                                       */
/*==============================================================*/
create index Coordinates_FK on JEE_Student (
   idcoordinator ASC
);

/*==============================================================*/
/* Table : JEE_SchoolMember                                     */
/*==============================================================*/
create table JEE_SchoolMember (
   idschmember      INTEGER   not null,
   fname    VARCHAR2(100)   not null,
   lname    VARCHAR2(100)   not null,
   tel      VARCHAR2(50)    not null,
   mail     VARCHAR2(100)   not null,
   usn      VARCHAR2(100)   not null,
   pwd      VARCHAR2(100)   not null,
   constraint PK_JEE_SchoolMember primary key (idschmember)
);

/*==============================================================*/
/* Table : JEE_Corporate                                        */
/*==============================================================*/
create table JEE_Corporate (
   ident            INTEGER       not null,
   entname          VARCHAR2(100)       not null,
   entspvname       VARCHAR2(100)       not null,
   entspvtel        VARCHAR2(100)       not null,
   entspvmail       VARCHAR2(100)       not null,
   entcountry       VARCHAR2(100)       not null,
   entcity          VARCHAR2(100)       not null,
   entstreet        VARCHAR2(100)       not null,
   entstreetnb      VARCHAR2(100)       not null,
   entzipcode       VARCHAR2(100)       not null,
   constraint PK_JEE_Corporate primary key (ident)
);
```

```
/*==============================================================*/
/* Table : JEE_Stage                                            */
/*==============================================================*/
create table JEE_Stage (
   idstage      INTEGER       not null,
   tfetheme     VARCHAR2(100) not null,
   datebegin    date,
   dateend      date,
   idstudent      INTEGER   not null,
   idschmember      INTEGER   not null,
   ident            INTEGER       not null,
   constraint PK_JEE_Stage primary key (idstage)
);
```

```
/*==============================================================*/
/* Index : PERFORM_FK                                           */
/*==============================================================*/
create index PERFORM_FK on JEE_Stage (
   idstudent ASC
);

/*==============================================================*/
/* Index : OVERSEES_FK                                          */
/*==============================================================*/
create index OVERSEES_FK on JEE_Stage (
   idschmember ASC
);

/*==============================================================*/
/* Index : TAKE_PLACE_FK                                        */
/*==============================================================*/
create index TAKE_PLACE_FK on JEE_Stage (
   ident ASC
);

/*==============================================================*/
/* CONSTRAINTS FOREIGN KEY                                      */
/*==============================================================*/

alter table JEE_Student
   add constraint FK_Student_COORDINATES_COORDINATOR foreign key (idcoordinator)
      references JEE_Coordinator (idcoordinator) on delete cascade;

alter table JEE_Stage
   add constraint FK_Stage_PERFORM_Student foreign key (idstudent)
      references JEE_Student (idstudent) on delete cascade;

alter table JEE_Stage
   add constraint FK_Stage_OVERSEES_SchoolMember foreign key (idschmember)
      references JEE_SchoolMember (idschmember) on delete cascade;

alter table JEE_Stage
   add constraint FK_Stage_TAKEPLACE_Corporate foreign key (ident)
      references JEE_Corporate (ident) on delete cascade;
```

```
/*==============================================================*/
/* AUTO INCREMENT                                               */
/*==============================================================*/
create sequence JEE_Coordinator_SEQ start with 1;

create or replace trigger JEE_Coordinator_T
before insert on JEE_Coordinator
for each row

begin
  select JEE_Coordinator_SEQ.nextval
  into   :new.idcoordinator
  from   dual;
end;
/

create sequence JEE_Student_SEQ start with 1;

create or replace trigger JEE_Student_T
before insert on JEE_Student
for each row

begin
  select JEE_Student_SEQ.nextval
  into   :new.idstudent
  from   dual;
end;
/

create sequence JEE_SchoolMember_SEQ START with 1;

create or replace trigger JEE_SchoolMember_T
before insert on JEE_SchoolMember
for each row

begin
  select JEE_SchoolMember_SEQ.nextval
  into   :new.idschmember
  from   dual;
end;
/
```

```
create sequence JEE_Corporate_SEQ START with 1;

create or replace trigger JEE_Corporate_T
before insert on JEE_Corporate
for each row

begin
  select JEE_Corporate_SEQ.nextval
  into   :new.ident
  from   dual;
end;
/

create sequence JEE_Stage_SEQ START with 1;

create or replace trigger JEE_Stage_T
before insert on JEE_Stage
for each row

begin
  select JEE_Stage_SEQ.nextval
  into   :new.idstage
  from   dual;
end;
/
```

Scripts PL/SQL

- Package :

Création d'un package « **ConnexionDB »** qui contient des fonctions d'authentifications des différents utilisateurs du programme

connexionCoordinator:	check la connexion d'un coordinateur.
connexionSchoolMember:	check la connexion d'un professeur.
connexionStudent:	check la connexion d'un étudiant.

Ces fonctions ont deux paramètre le login et le mot de passe et il renvoie un boolean pour les deux premiers et un number pour le dernier. Ce sont ces données qui seront traitées du code java.

Quant aux exceptions elles sont traitées dans chacune des fonctions. Lorsqu'une une exception ce déclenche, le programme pl/sql renvoie automatiquement l'exception. Celle-ci sera non seulement intercepté et gérée par le programme pl/sql mais le programme renvoie aussi l'exception côte java. Côté java, l'exception viendra sous forme SQLException et sera traitée.

Création d'un package « **AddData»** qui contient les différentes procédures d'ajout de données.

addCoordinator:	ajoute un coordinateur.
addEntreprise :	ajoute une entreprise.
addShoolMember:	ajoute un schoolMember.
addStudent:	ajoute un student
addStage:	ajoute un stage.

les exceptions dans cet package sont gérer par cette déclaration ci-dessous :

```
fk_violationException EXCEPTION;
PRAGMA EXCEPTION_INIT(fk_violationException,-2291);
```

S'il y a une violation des contraintes, pl/sql renvoie automatiquement une exception qui sera intercepté comme une SQLException en Java. Le message contenu dans l'exception sera traité côté java.

```
create or replace PACKAGE connexionDB
IS
  FUNCTION connexionCoordinator(login varchar2, pasword varchar2)
  RETURN BOOLEAN;

  FUNCTION connexionSchoolMember(login varchar2, pasword varchar2)
  RETURN BOOLEAN;

  FUNCTION connexionStudent(login varchar2, pasword varchar2)
  RETURN number ;
END connexionDB;
```

```
create or replace PACKAGE BODY connexionDB
IS
--------------- CONNECTION COORDINATOR--------------------------
  FUNCTION connexionCoordinator(login varchar2, pasword varchar2)
  RETURN BOOLEAN
  IS
    type coordinator is record(loginC JEE_COORDINATOR.USN%TYPE , pwdC JEE_COORDINATOR.PWD%type);
    type tab_coordinator is table of coordinator not null index by binary_integer;

    connexion boolean := false;
    t_coord tab_coordinator;
   error_connexion EXCEPTION;

  BEGIN
    select usn, pwd bulk collect into t_coord from JEE_COORDINATOR;
    for i IN 1..t_coord.count
      LOOP
       if t_coord(i).loginC = login and t_coord(i).pwdC = pasword then
          connexion := true;
       end if;
      END LOOP;

    if connexion = true then
      return true;
    else
      RAISE error_connexion;
    end if;

    Exception
      when error_connexion then
        DBMS_OUTPUT.PUT_LINE('ERROR LOGIN OR PASSWORD!');
        return false;
  END connexionCoordinator;
```

```
------------------------CONNECTION SChoolMemeber-----------------
FUNCTION connexionSchoolMember(login varchar2, pasword varchar2)
RETURN BOOLEAN
IS
  type schoolMemeber is record(loginC JEE_SCHOOLMEMBER.USN%TYPE , pwdC JEE_SCHOOLMEMBER.PWD%type);
 type tab_sm is table of schoolMemeber not null index by binary_integer;

 connexion boolean := false;
  t_sm tab_sm;
  error_connexion EXCEPTION;

BEGIN
  select usn, pwd bulk collect into t_sm from JEE_SCHOOLMEMBER;
  for i IN 1..t_sm.count
    LOOP
     if t_sm(i).loginC = login and t_sm(i).pwdC = pasword then
       connexion := true;
     end if;
    END LOOP;

  if connexion = true then
    return true;
  else
    RAISE error_connexion;
  end if;

  Exception
    when error_connexion then
      DBMS_OUTPUT.PUT_LINE('ERROR LOGIN OR PASSWORD!');
      return false;
 END connexionSchoolMember;
```

```
---------------------------- CONNECTION STUDENT--------------
  FUNCTION connexionStudent(login varchar2, pasword varchar2)
  RETURN number
 IS
   type student is record(loginC JEE_STUDENT.USN%TYPE , pwdC JEE_STUDENT.PWD%type);
   type tab_student is table of student not null index by binary_integer;

   connexion boolean := false;
   t_sm tab_student;
    error_connexion EXCEPTION;

 BEGIN
   select usn, pwd bulk collect into t_sm from JEE_STUDENT;
   for i IN 1..t_sm.count
    LOOP
       if t_sm(i).loginC = login and t_sm(i).pwdC = pasword then
        connexion := true;
       end if;
    END LOOP;

   if connexion = true then
     return 1;
   else
     RAISE error_connexion;
   end if;

   Exception
     when error_connexion then
       DBMS_OUTPUT.PUT_LINE('ERROR LOGIN OR PASSWORD!');
       return 0;
 END connexionStudent;

END connexionDB;
```

PacKage addData**

```
CREATE OR REPLACE PACKAGE AddData
AS
--procédures
procedure addCoordinator(nameC varchar2,login varchar2, pwd varchar2  ) ;

procedure addEntreprise(nameE varchar2,spvName varchar2,spvTel varchar2, spvMail varchar2,countryE varchar2,
           cityE  varchar2,streetE varchar2, streetNbE varchar2,   codeE varchar2) ;

procedure addShoolMember(fnameSm varchar2, lnameSm varchar2,tel varchar2,email varchar2,login varchar2,pasword varchar2) ;

procedure addStudent(fnameS varchar2,lnameS varchar2,tel varchar2,email varchar2,login varchar2,pasword varchar2,
                  section varchar2,catego varchar2,establissement varchar2, ville varchar2,
                  roude varchar2, roudenb number,code number,idcoord number ) ;

procedure addStage(tfe JEE_STAGE.TFETHEME%TYPE, dateB VARCHAR2, dateE varchar2, idStu number,idSchoolM number,idE number) ;

  --Exception
  fk_violationException EXCEPTION;
      PRAGMA EXCEPTION_INIT(fk_violationException,-2291);
END AddData;
/
```

```sql
--Corp du package et des procedures
CREATE OR REPLACE PACKAGE BODY AddData
IS
  fk_violationException EXCEPTION;
    PRAGMA EXCEPTION_INIT(fk_violationException,-2291);

  -- ADDCOORDINATOR-------------
 procedure addCoordinator(
      nameC varchar2,
      login varchar2,
      pwd varchar2
      )
as
begin
  INSERT INTO JEE_COORDINATOR (fname, usn, PWD)
  VALUES (nameC, login, pwd);
end addCoordinator;

--- ADDSCHOOLMEMBER--------------
 procedure addShoolMember(
      fnameSm varchar2,
      lnameSm varchar2,
      tel varchar2,
      email varchar2,
      login varchar2,
      pasword varchar2
      )
as
begin
  INSERT INTO JEE_SCHOOLMEMBER (fname, lname, tel, mail, usn, pwd)
  VALUES (fnameSm, lnameSm, tel, email, login, pasword );
end addShoolMember;
```

```
-- ADDSTUDENT--------------------
procedure addStudent(
      fnameS varchar2,
      lnameS varchar2,
      tel varchar2,
      email varchar2,
      login varchar2,
      pasword varchar2,
      section varchar2,
      catego varchar2,
      establissement varchar2,
      ville varchar2,
      roude varchar2,
      roudenb number,
      code number,
      idcoord number
      )
as
begin
  INSERT INTO JEE_Student (fname, lname, tel, mail, usn, pwd, sect, cat, estb, city, street, streetnb,zipcode, IDCOORDINATOR)
  VALUES (fnameS, lnameS, tel, email, login, pasword, section, catego, establissement, ville, roude, roudenb, code, idcoord )
end addStudent;

--- ADDENTREPRISE-----
procedure addEntreprise(
      nameE varchar2,
      spvName varchar2,
      spvTel varchar2,
      spvMail varchar2,
      countryE varchar2,
      cityE  varchar2,
      streetE varchar2,
      streetNbE varchar2,
      codeE varchar2
      )
as
begin
  INSERT INTO JEE_CORPORATE (ENTNAME, ENTSPVNAME, ENTSPVTEL, ENTSPVMAIL, ENTCOUNTRY, ENTCITY, ENTSTREET, ENTSTREETNB, ENTZIPC
  VALUES (nameE, spvName, spvTel, spvMail, countryE, cityE, streetE, streetNbE, codeE );
end addEntreprise;
```

```
--- ADDSTAGE------
  procedure addStage(
       tfe JEE_STAGE.TFETHEME%TYPE,
       dateB VARCHAR2,
       dateE varchar2,
       idStu number,
       idSchoolM number,
       idE number
       )
is
  vb_tfe JEE_STAGE.tfetheme%type;

begin

  INSERT INTO JEE_STAGE (TFETHEME, DATEBEGIN, DATEEND, IDSTUDENT, IDSCHMEMBER, IDENT)
  VALUES (tfe, dateB, dateE, idStu, idSchoolM, idE)
  returning tfetheme into vb_tfe;

  DBMS_OUTPUT.PUT_LINE('Theme: ' || vb_tfe);

end addStage;

END AddData;
/
```

Fonctions :

- **« getStudentID** » : cette fonction nous permet de retourner l'ID d'un étudiant dont son nom est passé en paramètre. Si aucun n'étudiant avec ce nom est trouvé une exception est déclenchée.

```
create or replace FUNCTION getStudentID(inUsn varchar2)
RETURN number
IS
  var_id JEE_Student.idStudent%type;

BEGIN
  select IDSTUDENT  into var_id from JEE_STUDENT
  where usn = inUsn;
  return var_id;

  Exception
    when NO_DATA_FOUND THEN
         DBMS_OUTPUT.PUT_LINE('pas d''etudiant trouvé avec ce login!');
         return 0;
END;
```

Procédures :

"***updateStudent*** ": cette procédure permet la mise à jour d'un étudiant ; il déclenche une exception lorsqu'il ne trouve pas l'étudiant correspondant.

```
create or replace procedure updateStudent(
      idS number,
      email varchar2,
      telS varchar2,
      roude varchar2,
      roudenb varchar2,
      ville varchar2,
      code varchar2,
      idCord number

      )
as
begin
  UPDATE JEE_STUDENT
  SET tel = telS,
      mail = email,
      city = ville,
      street = roude,
      streetnb = roudenb,
      zipcode = code,
      IDCOORDINATOR = idCord
    WHERE IDSTUDENT = idS;

    if SQL%found then
      DBMS_OUTPUT.PUT_LINE('The Data updated successfully and the number of lignes updated are: ' || SQL%ROWCOUNT);
    end if;

    Exception
      when NO_DATA_FOUND then
          DBMS_OUTPUT.PUT_LINE('Please the data you are looking for doesn''t exist!');
end updateStudent;
```

updateEntreprise ": cette procédure permet la mise à jour d'une entreprise.

```
create or replace procedure updateEntreprise(
      idE number,
      eName varchar2,
      spvName varchar2,
      spvTel varchar2,
      spvMail varchar2,
      eCountry varchar2,
      eCity varchar2,
      eStreet varchar2,
      eSreetnb varchar2,
      eZipcode varchar2
      )
as
begin
  UPDATE JEE_CORPORATE
  SET ENTNAME = eName,
      ENTSPVNAME = spvName,
      ENTSPVTEL = spvTel,
      ENTSPVMAIL = spvMail,
      ENTCOUNTRY = eCountry,
      ENTCITY   = eCity,
      ENTSTREET = eStreet,
      ENTSTREETNB = eSreetnb,
      ENTZIPCODE = eZipcode
    WHERE IDENT = idE;

    if SQL%found then
      DBMS_OUTPUT.PUT_LINE('The Data updated successfully and the number of lignes updated are: ' || SQL%ROWCOUNT);
    end if;

end updateEntreprise;
```

« ***updateStage*** » : cette procédure permet la mise à jour d'un stage ; vous remarquerez l'utilisation de la clause returning ainsi que le nombre de lignes qui ont été modifiées. Si aucune correspondance n'est retrouvée, une Exception de type « NO_DATA_FOUND » est déclenchée.

```
create or replace procedure updateStage(
      id_stage number,
      tfe varchar2,
      dateB varchar2,
      dateE varchar,
      id_student number,
      id_schoolMember number,
      id_ent number
      )
as
  vb_tfe JEE_STAGE.TFETHEME%type;
begin
  UPDATE JEE_STAGE
  SET TFETHEME = tfe,
      DATEBEGIN = dateB,
      DATEEND = dateE,
      IDSTUDENT = id_student,
      IDSCHMEMBER = id_schoolMember,
      IDENT = id_ent
    WHERE IDSTAGE = id_stage
    returning TFETHEME into vb_tfe;

    if SQL%found then
      DBMS_OUTPUT.PUT_LINE('The Data updated successfully and the number of lignes updated are: ' || SQL%ROWCOUNT);
      DBMS_OUTPUT.PUT_LINE('THEME: ' || vb_tfe);
    end if;

    Exception
     When NO_DATA_FOUND then
            DBMS_OUTPUT.PUT_LINE('Impossible to updtated ');
end updateStage;
```

« deleteStage » : cette dernière procédure permet la suppression d'un stage dont l'ID est passé en argument. Vous remarquerez l'utilisation d'un curseur.

```
create or replace procedure deleteStage(
      id_student number
      )
as
  cursor delete_Stage is select IDSTAGE , TFETHEME  from JEE_STAGE
  where IDSTUDENT = id_student;

begin
  for c in delete_Stage
    loop
       DELETE  JEE_STAGE
       commit;
    end loop;

    if SQL%found then
      DBMS_OUTPUT.PUT_LINE('The Data deleted successfully and the number of lignes updated are: ' || SQL%ROWCOUNT);
    end if;
    Exception
      When NO_DATA_FOUND then
              DBMS_OUTPUT.PUT_LINE('The data doesn''t exist !');
end deleteStage;
```