UNIVERSITÉ DE BOURGOGNE

UFR Sciences et Techniques

Mini Projet

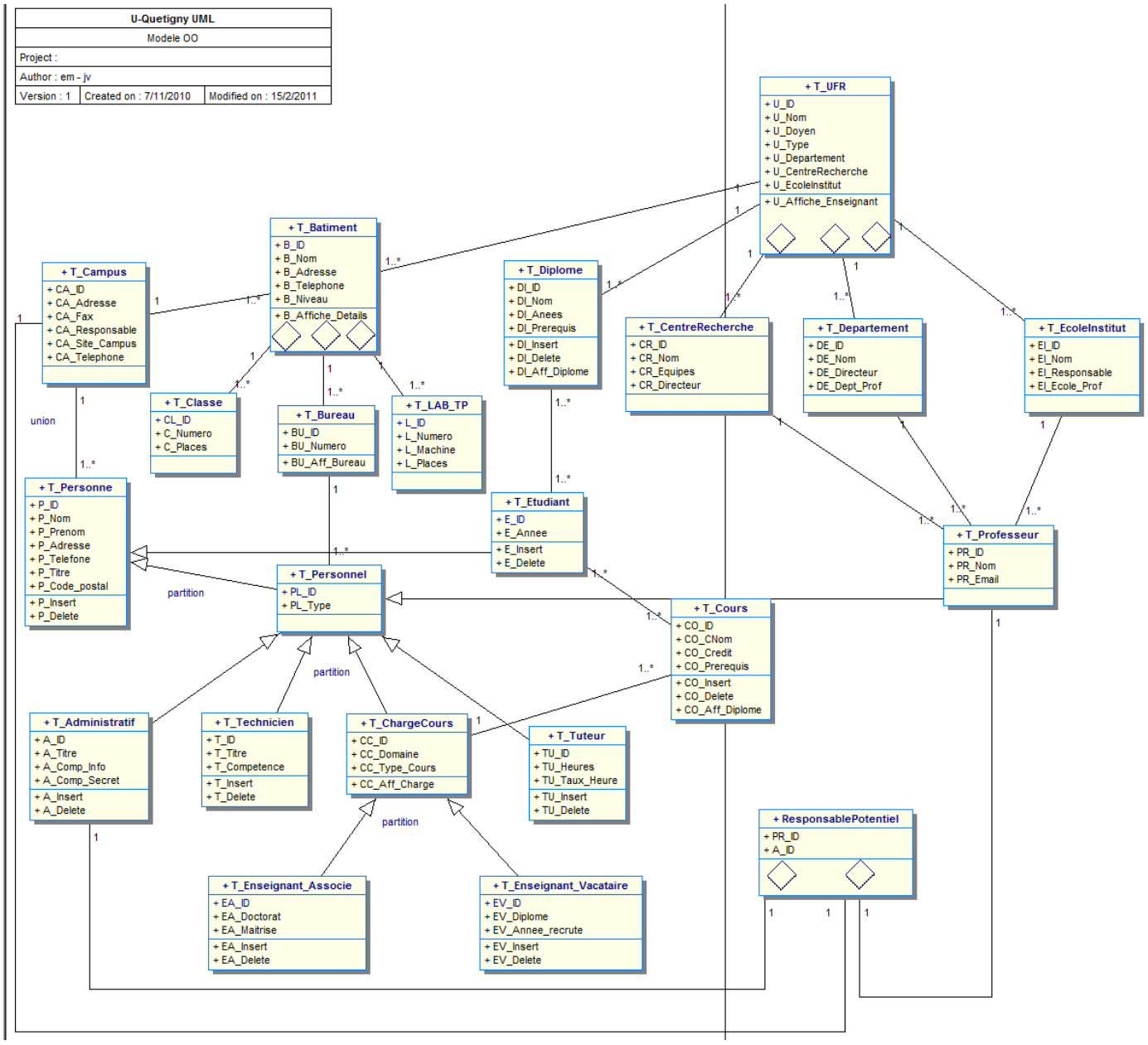
Cours: Système d'Information Orientés Objets

Présenté par: Elie Matta et al.

Table of Contents

1. Introduction

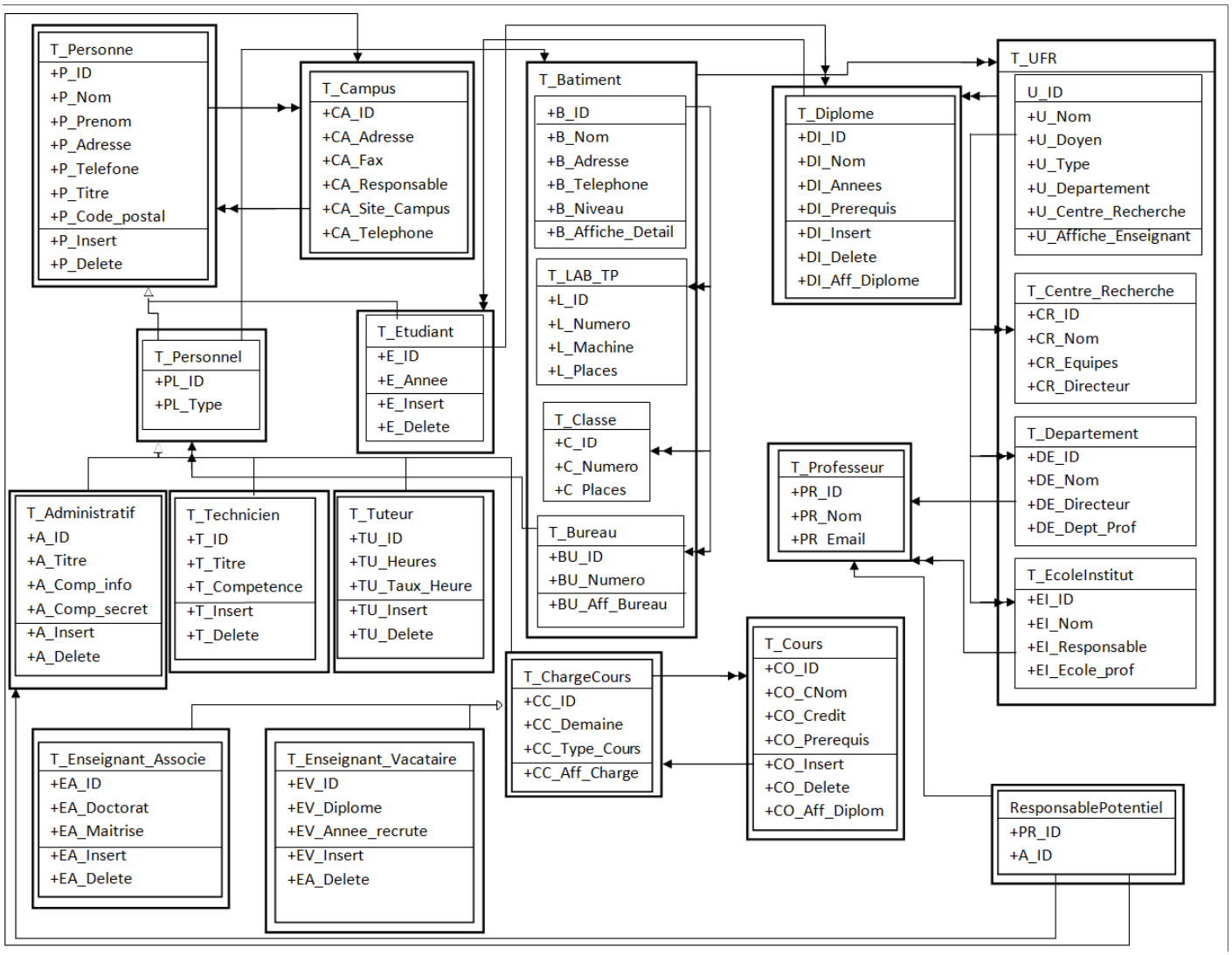
Following the schema below, we will proceed to implement this structure in an object-relational environment in Oracle using a web form:



2. Object-relational schema

We will draw the object-relational schema following the navigational model that consits of the following rules:

- The relations (1-1) will be presented by a single arrow (→)
- The relations (1--*) will be presented by two arrows (→→)
- We draw the arrow following the direction of the relation between the classes.
- Inheritance is presented by a white single arrow (—▷)
- Composition is presented by a diamond (◆)



3. Object-relational syntax

As we can see in the schema above, there are many types of notions about declaring classes, object and associations, therefore we will detail each implementation procedure in the oracle-relational concept using examples from our own implementation:

A. Types

In this level, we declare the objects without detailing any further constraints.

Example: Creating a type

```
CREATE OR REPLACE TYPE T_Personne_Type AS OBJECT (
      P_ID INTEGER,
      P_Nom VARCHAR2(100),
      P_Prenom VARCHAR2(100),
      P_Titre VARCHAR2(100),
      P_Adresse VARCHAR2(300)
)
/
```

B. Tables

Helps us storing the values and data by using CREATE TABLE syntax
These are storage structures that we use to store the value of objects and nested tables. Therefore we detail all the integrity constraints and triggers in this level.
It is essential here to mention that we have three kind of tables:

i. Relational table

Example: Relational table

```
create table user (
      Id not null,
      primary key(Id)
)
```

ii. Object-relational table

Example: Object relational table

```
CREATE TABLE T_Personne OF T_Personne_Type (
      PRIMARY KEY(P_ID)
      ) ;
```

iii. NESTED table

With this option, a table column will be able to contain almost any value.
We can use Varray to a fixed-size table, or a NESTED table without a size limit.
Usually Varray is more effective than nested tables.

Example: NESTED table - T_ProfesseursS

```
CREATE OR REPLACE TYPE T_Department_Type AS OBJECT    (
      DE_ID INTEGER,
      DE_Nom VARCHAR2(100),
      DE_Directeur VARCHAR2(100),
      DE_DeptProf VARCHAR2(100),
      T_ProfesseurS T_Professeur_Type,
      );
```

To implement this nested table, we should include it in the creation of the table:
Example: Table including a NESTED table

```
CREATE TABLE T_Departement OF T_Department_Type (
      PRIMARY KEY(DE_ID),
      T_ProfesseurS SCOPE IS T_Professeur
      )
      NESTED TABLE T_ProfesseurS STORE AS T_Prof;
```

Having this structure, we could insert into the table containing a nest table using the following syntax:

```
INSERT INTO T_Departement VALUES (DE_SEQUENCE.NEXTVAL, 'UFR-
IEM','KY',T_ProfesseursS());
INSERT INTO the (SELECT T_Departements FROM T_UFR WHERE U_ID = 1) SELECT
REF(D) FROM T_Departement D where DE_ID = 1;
```

C. Inheritance

The inheritance will help us modeling the relationship between the tables, this is a very essential relation between the tables since they rely on each others. In other words, in our project we have an inheritance between the T_Personnel table that permits us to create different kinds of personals using the UNDER syntax.

Example: Inheritance between T_Personnel and T_Tuteur

- T_Personnel_Type

```
CREATE OR REPLACE TYPE T_Personnel_Type UNDER T_Personne_Type (
      PL_Type VARCHAR2(100)
      )NOT INSTANTIABLE NOT FINAL;
/

CREATE TABLE T_Personnel OF T_Personnel_Type;
```

- T_Tuteur_Type

```
CREATE TYPE T_Tuteur_Type UNDER T_Personnel_Type (
      TU_ID INTEGER,
      TU_Heure INTEGER,
      TU_TauxHeures VARCHAR2(100)
      )INSTANTIABLE FINAL;
/
CREATE TABLE T_Tuteur OF T_Tuteur_Type(PRIMARY KEY(TU_ID));
```

It is essential here to explain that by default, the object type is FINAL which means that he could not have an UNDER type.

D. Reference

The **REF** keyword is a logical reference to an object from a table of type object. It relies on the use of indexes.
To use this reference, we first declare it in the TYPE using the REF keyword, the we decide what scope it should in using the SCOPE IS syntax shown below:

Example: Reference between the tables

- Type: T_EcoleInstitut_Type

```
CREATE OR REPLACE TYPE T_EcoleInstitut_Type AS OBJECT (
      EI_ID INTEGER,
      EI_Nom VARCHAR2(100),
      EI_Directeur VARCHAR2(100),
      EI_EcoleProf VARCHAR2(100),
      T_ProfesseurS REF T_Professeur_Type,
      T_UFR_EI T_UFRs_Type
      );

CREATE OR REPLACE TYPE T_EcoleInstitut_Ref_Type AS OBJECT (
      T_EcoleInstitut REF T_EcoleInstitut_Type
      );
```

- Table: T_EcoleInstitut

```
CREATE TABLE T_EcoleInstitut OF T_EcoleInstitut_Type (
      PRIMARY KEY(EI_ID),
      T_ProfesseurS SCOPE IS T_Professeur
      )
      NESTED TABLE T_UFR_EI STORE AS EI_UFR;
```

To use this reference, we will have to select it using an embedded select query from the main table as shown below:

```
INSERT INTO T_EcoleInstitut VALUES (DE_SEQUENCE.NEXTVAL, 'Mirande','Dr
Toffee','O.B',T_ProfesseursS());
```

```
INSERT INTO the (SELECT T_EcoleInstitut FROM T_UFR WHERE U_ID = 1) SELECT
REF(E) FROM T_EcoleInstitut E where EI_ID = 1;
```

We can notice the limitation with the references that it could not be a primary key or a unique key. It's use is also limited on navigation between tables without entering the check list of unique keys.

E. NESTED TABLES or REF?

From the examples given above, we concluded that the NESTED TABLES is far more extensible and beneficial for navigating from a table to another. NESTD TABLES also give us the advantage to dynamically create tables embedded in each other's so it permits us to extend each table as much as we want.

F. Methods

We could declare methods using the MEMBER syntax that indicates that this particular method is an instance for this method.
Declaration for methods are held in the types, while the implementation is detailed in the BODY.

Example: Using methods

- Declaration

```
CREATE OR REPLACE TYPE T_Batiment_Type AS OBJECT (
      B_ID INTEGER,
      B_Nom VARCHAR2(100),
      B_Adresse VARCHAR2(300),
      B_Telephone INTEGER,
      B_Niveau INTEGER,
      T_CampusS T_CampusS_Type,
      T_UFR_B T_UFRs_Type,
      T_BureauS REF T_Bureau_Type,
      T_ClasseS REF T_Classe_Type,
      T_Lab_TpS REF T_Lab_Tp_Type,
      MEMBER FUNCTION afficheDetails RETURN VARCHAR2
      );
```

- Implementation

```
CREATE OR REPLACE BODY T_Batiment_Type AS
      MEMBER FUNCTION afficheDetails RETURN VARCHAR2 IS resultat
VARCHAR2(6000);
      BEGIN
      SELECT B.B_ID||' '||BU.Numero||' '||CL.Numero||' '||CL.Places||'
'||L.Machines
      INTO resultat FROM T_Batiment_Type B, T_Bureau BU, T_Classe CL,
T_Lab_Tp L
```

```
      WHERE B.B_ID = SELF.B_ID;
      RETURN resultat;
      END;
END;
```

G. Triggers

We used triggers to help us for the insertion and manipulation of the tables. We further used sequences to autoincrement the primary keys while navigating the tables.

Example: Triggers and sequences

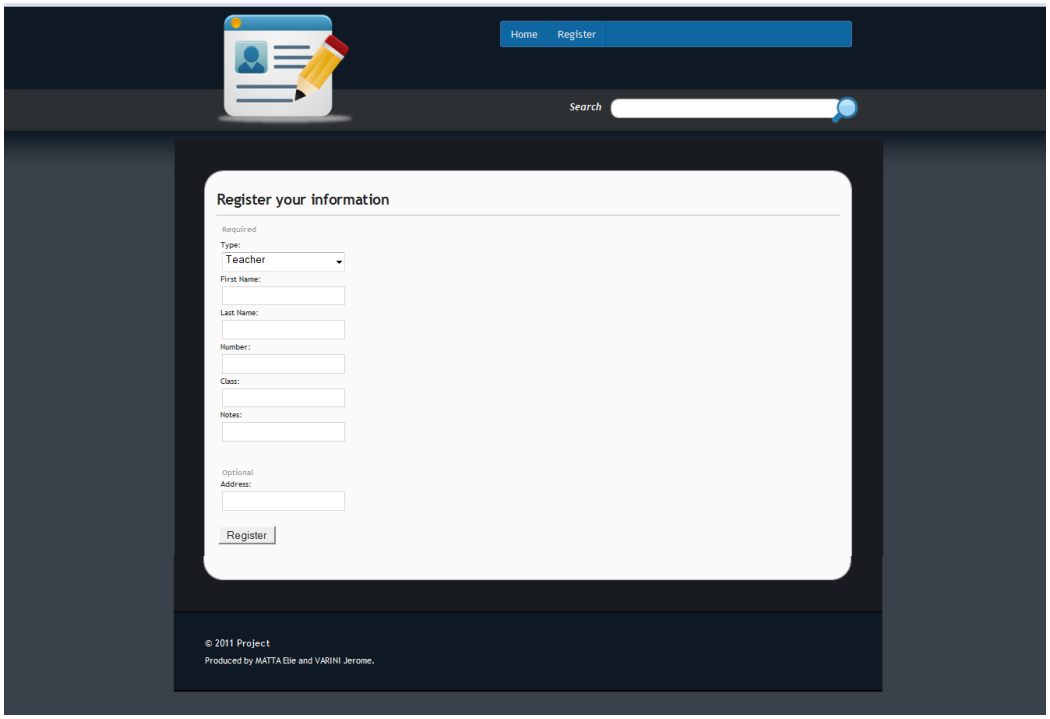
```
CREATE SEQUENCE U_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER U_Trigger BEFORE INSERT ON T_UFR
FOR EACH ROW
      BEGIN
            SELECT U_Sequence.NEXTVAL INTO :NEW.U_ID FROM DUAL;
      END;
```

3. Web form

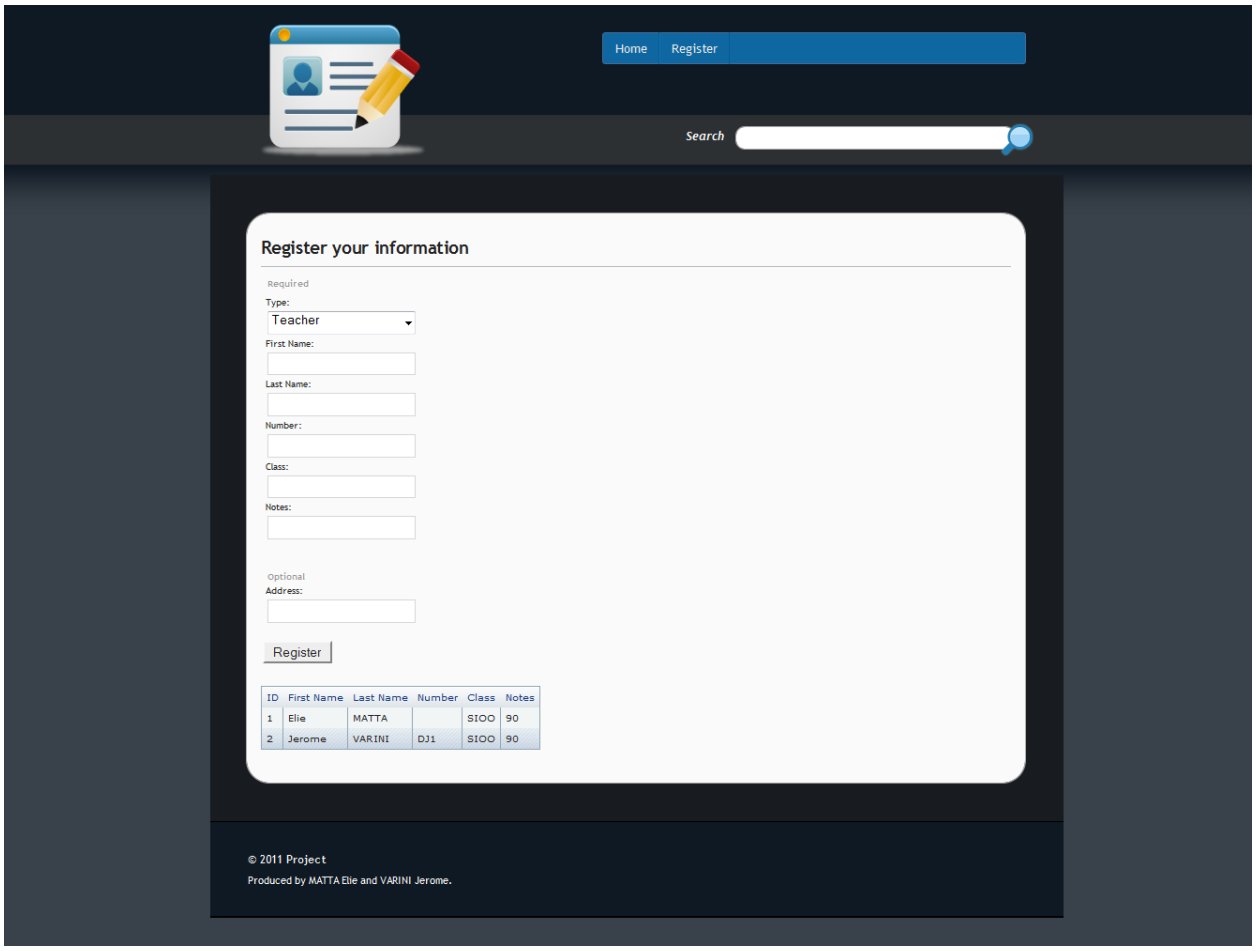
To implement our project, we created a web form for the use of personals, therefore we used the combination of the following features:

- ODP.NET
- Oracle database
- ASP.NET

The following form presents a registration area by type, therefore the user chooses if he was a student, teacher,... and according to this choice, he will further submit and enter the personals list in the oracle database.



And after we insert the data, we could see the result as follows:



4. Source code

A. Web form

i. Connection class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;


public class ConnectionParams
{
    public static string Datasource = "ufrsciencestech.u-
bourgogne.fr:25559/eluard/ens098";
    public static string Username = "em159989";      //Username
    public static string Password = "em159989";      //Password
}
```

ii. Form

```
using System;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

public partial class _Default : System.Web.UI.Page
{

    private void insertPersonnel(string type, string fn, string ln, string c, string
note, string address)
    {
        //To fill DataSet from datasource
        OracleDataAdapter personnelAdapter;

        //Represents Stored Procedure to execute against a data source
        OracleCommand personnelCmd;

        DataSet personnelsDataSet;

        try
        {
```

```
            personnelCmd = new OracleCommand();


            personnelCmd.CommandText = "ODPNet.UFR";

            personnelCmd.CommandType = CommandType.StoredProcedure;

            personnelCmd.Connection = conn;


            personnelCmd.Parameters.Add("T_Personnel", OracleDbType.RefCursor,
                           DBNull.Value, ParameterDirection.Output);

            personnelCmd.Parameters.Add("T_Admin", OracleDbType.RefCursor,
                      DBNull.Value, ParameterDirection.Output);

            personnelAdapter = new OracleDataAdapter(personnelCmd);

            personnelsDataSet = new DataSet("dsPersonnel");



            personnelAdapter.Fill(personnelsDataSet, "personnels");


        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }


    private Boolean getDBConnection()
    {
        try
        {
            string connectionString =
                "User Id=" + ConnectionParams.Username +
                ";Password=" + ConnectionParams.Password +
                ";Data Source=" + ConnectionParams.Datasource;
            conn = new OracleConnection(connectionString);
            conn.Open();

            return true;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
            return false;
        }
    }
}
```

```
protected void Page_Load(object sender, EventArgs e)
{
    DataSet dsPersonnel = new DataSet();

    DataTable dataTable = new DataTable();

    DataColumn dataColumn = new DataColumn();

    dataColumn.AllowDBNull = false;
    dataColumn.AutoIncrement = true;
    dataColumn.AutoIncrementSeed = 1;
    dataColumn.AutoIncrementStep = 1;
    dataColumn.ColumnName = "ID";
    dataColumn.DataType = System.Type.GetType("System.Int32");
    dataColumn.Unique = true;
    dataTable.Columns.Add(dataColumn);

    dataColumn = new DataColumn();
    dataColumn.ColumnName = "firstName";
    dataColumn.DataType = System.Type.GetType("System.String");
    dataTable.Columns.Add(dataColumn);

    dataColumn = new DataColumn();
    dataColumn.ColumnName = "lastName";
    dataColumn.DataType = System.Type.GetType("System.String");
    dataTable.Columns.Add(dataColumn);

    dataColumn = new DataColumn();
    dataColumn.ColumnName = "number";
    dataColumn.DataType = System.Type.GetType("System.String");
    dataTable.Columns.Add(dataColumn);

    dataColumn = new DataColumn();
    dataColumn.ColumnName = "class";
    dataColumn.DataType = System.Type.GetType("System.String");
    dataTable.Columns.Add(dataColumn);

    dataColumn = new DataColumn();
    dataColumn.ColumnName = "notes";
    dataColumn.DataType = System.Type.GetType("System.Double");
    dataTable.Columns.Add(dataColumn);

    DataRow dataRow = dataTable.NewRow();

    dataRow["firstName"] = "Elie";
    dataRow["lastName"] = "MATTA";
    dataRow["class"] = "SIOO";
    dataRow["notes"] = 90;
    dataTable.Rows.Add(dataRow);


    dataRow = dataTable.NewRow();
```

```
        dataRow["firstName"] = "Jerome";
        dataRow["lastName"] = "VARINI";
        dataRow["number"] = "DJ1";
        dataRow["class"] = "SIOO";
        dataRow["notes"] = 90;
        dataTable.Rows.Add(dataRow);

        GridView1.DataSource = dataTable;
        GridView1.DataBind();

        DataTable dataTableType = new DataTable();

        DataColumn dataColumnType = new DataColumn();

        dataColumnType.AllowDBNull = false;
        dataColumnType.AutoIncrement = true;
        dataColumnType.AutoIncrementSeed = 1;
        dataColumnType.AutoIncrementStep = 1;
        dataColumnType.ColumnName = "TypeID";
        dataColumnType.DataType = System.Type.GetType("System.Int32");
        dataColumnType.Unique = true;
        dataTableType.Columns.Add(dataColumnType);

        dataColumnType = new DataColumn();
        dataColumnType.ColumnName = "TypeName";
        dataColumnType.DataType = System.Type.GetType("System.String");
        dataTableType.Columns.Add(dataColumnType);

        DataRow dataRowType = dataTableType.NewRow();

        dataRowType["TypeName"] = "Teacher";
        dataTableType.Rows.Add(dataRowType);


        dataRowType = dataTableType.NewRow();
        dataRowType["TypeName"] = "Student";
        dataTableType.Rows.Add(dataRowType);

        DropDownList1.DataSource = dataTableType;
        DropDownList1.DataBind();
    }
    protected void btnRegister_Click(object sender, EventArgs e)
    {
        insertPersonnel(TypeName.selectedOption, firstName.text, lastName.text,
studentNumber.text, studentClass.text, studentNotes.text, address.text);
    }
```

```
CREATE TYPE T_UFR_Type;
CREATE TYPE T_UFR_Ref_Type;
CREATE TYPE T_UFRs_Type;
```

B. Database

```
CREATE TYPE T_CentreRecherche_Type;
CREATE TYPE T_CentreRecherche_Ref_Type;
CREATE TYPE T_CentreRechercheS_Type;
CREATE TYPE T_Department_Type;
CREATE TYPE T_Departement_Ref_Type;
CREATE TYPE T_DepartementS_Type;
CREATE TYPE T_EcoleInstitut_Type;
CREATE TYPE T_EcoleInstitut_Ref_Type;
CREATE TYPE T_EcoleInstitutS_Type;
CREATE TYPE T_Batiment_Type;
CREATE TYPE T_Batiment_Ref_Type;
CREATE TYPE T_BatimentS_Type;
CREATE TYPE T_Bureau_Type;
CREATE TYPE T_Bureau_Ref_Type;
CREATE TYPE T_BureauS_Type;
CREATE TYPE T_Personnel_Type;
CREATE TYPE T_Diplome_Type;
CREATE TYPE T_Etudiant_Type;
CREATE TYPE Etudiant_Diplome_Type;
CREATE TYPE T_Cours_Type;
CREATE TYPE Etudiant_Cours_Type;
CREATE TYPE T_Campus_Type;
CREATE TYPE T_Campus_Ref_Type;
CREATE TYPE T_CampusS_Type;
CREATE TYPE T_Admin_Type;
CREATE TYPE T_Professeur_Type;
CREATE TYPE T_Classe_Type;
CREATE TYPE T_Lab_Tp_Type;
CREATE TYPE T_Personne_Type;
CREATE TYPE T_Technicien_Type;
CREATE TYPE T_ChargeCours_Type;
CREATE TYPE T_Tuteur_Type;
CREATE TYPE T_Enseignant_Associate_Type;
CREATE TYPE T_Enseignant_Vacataire_Type;
CREATE TYPE T_ResponsablePotentiel_Type;
CREATE TYPE T_ResponsablePotentiel_Ref_Type;
CREATE TYPE T_ResponsablePotentielS_Type;

CREATE OR REPLACE TYPE T_UFR_Type AS OBJECT (
      U_ID INTEGER,
      U_Nom VARCHAR2(100),
      U_Doyen VARCHAR2(100),
      U_Type VARCHAR2(10),
      T_CentreRechercheS REF T_CentreRecherche_Type,
      T_DepartementS REF T_Department_Type,
      T_EcoleInstitutS REF T_EcoleInstitut_Type,
      T_BatimentS REF T_Batiment_Type,
      T_DipolomeS REF T_Diplome_Type,
      MEMBER FUNCTION afficheEnseignant RETURN VARCHAR2
      );

CREATE OR REPLACE TYPE T_UFR_Ref_Type AS OBJECT (
      T_UFR REF T_UFR_Type
```

```
	);

CREATE OR REPLACE TYPE T_UFRs_Type AS TABLE OF T_UFR_Ref_Type;

CREATE TABLE T_UFR OF T_UFR_Type (
	PRIMARY KEY(U_ID),
	T_CentreRechercheS SCOPE IS T_CentreRecherche,
	T_BatimentS SCOPE IS T_Batiment,
	T_DepartementS SCOPE IS T_Departement,
	T_EcoleInstitutS SCOPE is T_EcoleInstitut,
	T_DiplomeS SCOPE IS T_Diplome
	);

CREATE SEQUENCE U_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER U_Trigger BEFORE INSERT ON T_UFR
FOR EACH ROW
	BEGIN
		SELECT U_Sequence.NEXTVAL INTO :NEW.U_ID FROM DUAL;
	END;


CREATE OR REPLACE BODY T_UFR_Type AS
	MEMBER FUNCTION afficheEnseignant RETURN VARCHAR2 IS resultat
VARCHAR2(6000);
	BEGIN
	SELECT U.U_ID||' '||EI.EcoleProf||' '||DE.DeptProf INTO resultat FROM
UFR U, T_EcoleInstitutS EI, T_DepartementS DE,
	WHERE U.U_ID = SELF.U_ID;
	RETURN resultat;
	END;
END;



CREATE OR REPLACE TYPE T_CentreRecherche_Type AS OBJECT (
	CR_ID INTEGER,
	CR_Nom VARCHAR2(100),
	CR_Directeur VARCHAR2(100),
	CR_Equipes VARCHAR2(100),
	T_UFR_CR T_UFRs_Type
	);

CREATE OR REPLACE TYPE T_CentreRecherche_Ref_Type AS OBJECT (
	T_CentreRecherche REF T_CentreRecherche_Type
	);

CREATE OR REPLACE TYPE T_CentreRechercheS_Type AS TABLE OF
T_CentreRecherche_Ref_Type;


CREATE TABLE T_CentreRecherche OF T_CentreRecherche_Type (
	PRIMARY KEY(CR_ID),
	T_ProfesseurS SCOPE IS T_Professeur
```

```
	)
	NESTED TABLE T_UFR_CR STORE AS CR_UFR;


CREATE SEQUENCE CR_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER CR_Trigger BEFORE INSERT ON T_CentreRecherche
FOR EACH ROW
	BEGIN
		SELECT CR_Sequence.NEXTVAL INTO :NEW.CR_ID FROM DUAL;
	END;



CREATE OR REPLACE TYPE T_Department_Type AS OBJECT    (
	DE_ID INTEGER,
	DE_Nom VARCHAR2(100),
	DE_Directeur VARCHAR2(100),
	DE_DeptProf VARCHAR2(100),
	T_ProfesseurS REF T_Professeur_Type,
	T_UFR_DE T_UFRs_Type
	);

CREATE OR REPLACE TYPE T_Departement_Ref_Type AS OBJECT (
	T_Departement REF T_Department_Type
	);
CREATE OR REPLACE TYPE T_DepartementS_Type AS TABLE OF
T_Departement_Ref_Type;

CREATE TABLE T_Departement OF T_Department_Type (
	PRIMARY KEY(DE_ID),
	T_ProfesseurS SCOPE IS T_Professeur
	)
	NESTED TABLE T_UFR_DE STORE AS DE_UFR;


CREATE SEQUENCE DE_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER DE_Trigger BEFORE INSERT ON T_Departement
FOR EACH ROW
	BEGIN
		SELECT DE_Sequence.NEXTVAL INTO :NEW.DE_ID FROM DUAL;
	END;



CREATE OR REPLACE TYPE T_EcoleInstitut_Type AS OBJECT (
	EI_ID INTEGER,
	EI_Nom VARCHAR2(100),
	EI_Directeur VARCHAR2(100),
```

```
	EI_EcoleProf VARCHAR2(100),
	T_ProfesseurS REF T_Professeur_Type,
	T_UFR_EI T_UFRs_Type
	);

CREATE OR REPLACE TYPE T_EcoleInstitut_Ref_Type AS OBJECT (
	T_EcoleInstitut REF T_EcoleInstitut_Type
	);
CREATE OR REPLACE TYPE T_EcoleInstitutS_Type AS TABLE OF
T_EcoleInstitut_Ref_Type;

CREATE TABLE T_EcoleInstitut OF T_EcoleInstitut_Type (
	PRIMARY KEY(EI_ID),
	T_ProfesseurS SCOPE IS T_Professeur
	)
	NESTED TABLE T_UFR_EI STORE AS EI_UFR;


CREATE SEQUENCE EI_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER EI_Trigger BEFORE INSERT ON T_EcoleInstitut
FOR EACH ROW
	BEGIN
		SELECT EI_Sequence.NEXTVAL INTO :NEW.EI_ID FROM DUAL;
	END;




CREATE OR REPLACE TYPE T_Batiment_Type AS OBJECT (
	B_ID INTEGER,
	B_Nom VARCHAR2(100),
	B_Adresse VARCHAR2(300),
	B_Telephone INTEGER,
	B_Niveau INTEGER,
	T_CampusS T_CampusS_Type,
	T_UFR_B T_UFRs_Type,
	T_BureauS REF T_Bureau_Type,
	T_ClasseS REF T_Classe_Type,
	T_Lab_TpS REF T_Lab_Tp_Type,
	MEMBER FUNCTION afficheDetails RETURN VARCHAR2
	);

CREATE OR REPLACE TYPE T_Batiment_Ref_Type AS OBJECT (
	T_Batiment REF T_Batiment_Type
	);

CREATE OR REPLACE TYPE T_BatimentS_Type AS TABLE OF T_Batiment_Ref_Type;

CREATE TABLE T_Batiment OF T_Batiment_Type (
	PRIMARY KEY(B_ID),
	T_BreauS SCOPE IS T_Bureau,
	T_ClasseS SCOPE IS T_Classe,
```

```
      T_Lab_TpS SCOPE IS T_Lab_Tp
      )
      NESTED TABLE T_CampusS STORE AS B_Campus,
      NESTED TABLE T_UFR_B STORE AS B_UFR;

CREATE SEQUENCE B_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER B_Trigger BEFORE INSERT ON T_Batiment
FOR EACH ROW
      BEGIN
            SELECT B_Sequence.NEXTVAL INTO :NEW.B_ID FROM DUAL;
      END;

CREATE OR REPLACE BODY T_Batiment_Type AS
      MEMBER FUNCTION afficheDetails RETURN VARCHAR2 IS resultat
VARCHAR2(6000);
      BEGIN
      SELECT B.B_ID||' '||BU.Numero||' '||CL.Numero||' '||CL.Places||'
'||L.Machines
      INTO resultat FROM T_Batiment_Type B, T_Bureau BU, T_Classe CL,
T_Lab_Tp L
      WHERE B.B_ID = SELF.B_ID;
      RETURN resultat;
      END;
END;




CREATE OR REPLACE TYPE T_Bureau_Type AS OBJECT (
      BU_ID INTEGER,
      BU_Numero INTEGER,
      BU_Telephone INTEGER,
      T_PersonnelS REF T_Personnel_Type,
      T_Batiment_BU T_BatimentS_Type,
      MEMBER FUNCTION afficheBureau RETURN IN VARCHAR2
      );

CREATE OR REPLACE TYPE T_Bureau_Ref_Type AS OBJECT (
      T_Bureau REF T_Bureau_Type
      );

CREATE OR REPLACE TYPE T_BureauS_Type AS TABLE OF T_Bureau_Ref_Type;


CREATE TABLE T_Bureau OF T_Bureau_Type (
      PRIMARY KEY(BU_Numero),
      T_PersonnelS SCOPE IS T_Personnel
      )
      NESTED TABLE T_Batiment_BU STORE AS BU_Batiment;
```

```
CREATE SEQUENCE BU_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER BU_Trigger BEFORE INSERT ON T_Bureau
FOR EACH ROW
	BEGIN
		SELECT BU_Sequence.NEXTVAL INTO :NEW.BU_ID FROM DUAL;
	END;

	CREATE OR REPLACE BODY T_Bureau_Type AS
	MEMBER FUNCTION afficheBureau RETURN VARCHAR2 IS resultat
VARCHAR2(6000);
	BEGIN
	SELECT BU_Numero||' '||BU_Telephone INTO resultat FROM T_Bureau_Type BU
	WHERE BU.BU_ID = SELF.BU_ID;
	RETURN resultat;
	END;
END;




CREATE OR REPLACE TYPE T_Personnel_Type UNDER T_Personne_Type (
	PL_Type VARCHAR2(100)
	T_Bureau_PL t_BureauS_type
	)
	NOT INSTANTIABLE, NOT FINAL;


CREATE TABLE T_Personnel OF T_Personnel_Type
	NESTED TABLE T_Bureau_PL STORE AS PL_Bureau;



CREATE OR REPLACE TYPE T_Diplome_Type AS OBJECT (
	DI_ID INTEGER,
	DI_Nom VARCHAR2(100),
	DI_NAnnees INTEGER,
	DI_Prerequis VARCHAR2(100),
	T_UFR_DI T_UFRs_Type
	MEMBER FUNCTION afficheDiplome RETURN VARCHAR2,
	MEMBER PROCEDURE DI_Insert(
		ID IN INTEGER,
		Nom IN VARCHAR2,
		Annees IN VARCHAR2,
		Prerequis IN VARCHAR2
		),
	MEMBER PROCEDURE DI_Delete
	);

CREATE TABLE T_Diplome OF T_Diplome_Type (
	PRIMARY KEY(DI_ID)
```

```
	)
	NESTED TABLE T_UFR_DI STORE AS DI_UFR;

CREATE SEQUENCE DI_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER DI_Trigger BEFORE INSERT ON T_Diplome
FOR EACH ROW
	BEGIN
		SELECT DI_Sequence.NEXTVAL INTO :NEW.DI_ID FROM DUAL;
	END;


CREATE OR REPLACE BODY T_Diplome_Type AS MEMBER PROCEDURE DI_Insert (
	ID IN INTEGER,
	Nom IN VARCHAR2,
	Annees IN VARCHAR2,
	Prerequis IN VARCHAR2
	)
	BEGIN
	INSERT INTO T_Diplome_Type ( DI_ID, DI_Nom, DI_NAnnees, DI_Prerequis )
	VALUES ( ID, Nom, Annees, Prerequis );
	END;


	MEMBER PROCEDURE DI_Delete
	BEGIN
	DELETE FROM T_Diplome_Type WHERE DI_ID = SELF.DI_ID;
	END;

	MEMBER FUNCTION afficheDiplome RETURN VARCHAR2 IS
	resultat VARCHAR2(6000);
	BEGIN
	SELECT DI_ID||' '||DI_Nom||' '||DI_NAnnees||' '||DI_Prerequis INTO
resultat
	FROM T_Diplome_Type D
	WHERE D.DI_ID = SELF.DI_ID;
	RETURN resultat;
	END;
END;



CREATE OR REPLACE TYPE T_Etudiant_Type AS OBJECT (
	E_ID INTEGER,
	E_Annee INTEGER,
	MEMBER PROCEDURE E_Insert (
	Annee IN VARCHAR2
	),
	MEMBER PROCEDURE E_Delete); INSTANTIABLE FINAL ;

CREATE TABLE T_Etudiant OF T_Etudiant_Type (PRIMARY KEY(E_ID));
```

```
CREATE SEQUENCE E_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER E_Trigger BEFORE INSERT ON T_Etudiant
FOR EACH ROW
      BEGIN
            SELECT E_Sequence.NEXTVAL INTO :NEW.E_ID FROM DUAL;
      END;

CREATE OR REPLACE BODY T_Etudiant_Type AS MEMBER PROCEDURE E_Insert (
      Annee IN INTEGER
      )
      BEGIN
      INSERT INTO T_Etudiant_Type( E_Annee ) VALUES ( Annee );
      END;


MEMBER PROCEDURE E_Delete
      BEGIN
      DELETE FROM T_Etudiant_Type WHERE E_Annee = SELF.E_Annee;
      END;
END;


CREATE TYPE T_Admin_Type UNDER T_Personnel_Type(
      A_ID INTEGER,
      A_Titre VARCHAR2(100),
      A_CompInfo VARCHAR2(100),
      A_CompSecret VARCHAR2(100)
      )INSTANTIABLE FINAL;
/



CREATE TABLE T_Admin OF T_Admin_Type (
      PRIMARY KEY(A_ID)
      );

CREATE SEQUENCE A_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER A_Trigger BEFORE INSERT ON T_Admin
FOR EACH ROW
      BEGIN
            SELECT A_Sequence.NEXTVAL INTO :NEW.A_ID FROM DUAL;
      END;


CREATE TYPE T_Technicien_Type UNDER T_Personnel_Type (
      T_ID INTEGER,
      T_Titre VARCHAR2(100),
      T_Competence VARCHAR2(100)
      )INSTANTIABLE FINAL;
/
CREATE TABLE T_Technicien OF T_Technicien_Type(PRIMARY KEY(T_ID));
```

```
CREATE SEQUENCE T_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER T_Trigger BEFORE INSERT ON T_Technicien
FOR EACH ROW
      BEGIN
            SELECT T_Sequence.NEXTVAL INTO :NEW.T_ID FROM DUAL;
      END;

CREATE TYPE T_Tuteur_Type UNDER T_Personnel_Type (
      TU_ID INTEGER,
      TU_Heure INTEGER,
      TU_TauxHeures VARCHAR2(100)
      )INSTANTIABLE FINAL;
/
CREATE TABLE T_Tuteur OF T_Tuteur_Type(PRIMARY KEY(TU_ID));

CREATE SEQUENCE TU_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER TU_Trigger BEFORE INSERT ON T_Tuteur
FOR EACH ROW
      BEGIN
            SELECT TU_Sequence.NEXTVAL INTO :NEW.TU_ID FROM DUAL;
      END;

CREATE TYPE T_ChargeCours_Type UNDER T_Personnel_Type (
      CC_ID INTEGER,
      CC_Domaine VARCHAR2(100)
      ) NOT INSTANTIABLE NOT FINAL;
/
CREATE TABLE T_ChargeCours OF T_ChargeCours_Type(PRIMARY KEY(CC_ID));

CREATE SEQUENCE CC_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER CC_Trigger BEFORE INSERT ON T_ChargeCours
FOR EACH ROW
      BEGIN
            SELECT CC_Sequence.NEXTVAL INTO :NEW.CC_ID FROM DUAL;
      END;



CREATE TYPE T_Enseignant_Vacataire_Type UNDER T_ChargeCours_Type (
      EV_ID INTEGER,
      EV_Diplome VARCHAR2(100),
      EV_AnneeRecrute INTEGER
      )INSTANTIABLE FINAL;
/

CREATE TABLE T_Enseignant_Vacataire OF T_Enseignant_Vacataire_Type (
      PRIMARY KEY(EV_ID)
      );
```

```
CREATE SEQUENCE EV_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER EV_Trigger BEFORE INSERT ON T_Enseignant_Vacataire
FOR EACH ROW
	BEGIN
		SELECT EV_Sequence.NEXTVAL INTO :NEW.EV_ID FROM DUAL;
	END;



CREATE OR REPLACE TYPE Etudiant_Diplome_Type AS OBJECT	(
	Etudiant_ED REF T_Etudiant_Type,
	Diplome_ED REF T_Diplome_Type);


CREATE TABLE Etudiant_Diplome OF Etudiant_Diplome_Type (
	Etudiant_ED SCOPE IS T_Etudiant,
	Diplome_ED SCOPE IS T_Diplome
	);




CREATE OR REPLACE TYPE T_Cours_Type AS OBJECT (
	CO_ID INTEGER,
	CO_CNom VARCHAR2(100),
	CO_Credit INTEGER,
	CO_Prerequis VARCHAR2(100),
	T_ChargeCours_CO T_ChargeCoursS_Type
	),
	MEMBER PROCEDURE CO_Insert(
	ID IN INTEGER,
	CNom IN VARCHAR2,
	Credit IN INTEGER,
	Prerequis IN VARCHAR2)
	MEMBER PROCEDURE CO_Delete);

CREATE TABLE T_Cours OF T_Cours_Type (
	PRIMARY KEY(CO_ID)
	)
NESTED TABLE T_ChargeCours_CO STORE AS CO_ChargeCours;

CREATE SEQUENCE CO_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER CO_Trigger BEFORE INSERT ON T_Cours
FOR EACH ROW
	BEGIN
		SELECT CO_Sequence.NEXTVAL INTO :NEW.CO_ID FROM DUAL;
	END;

CREATE OR REPLACE BODY T_Cours_Type AS
	MEMBER PROCEDURE CO_Insert (
		ID IN INTEGER,
```

```
            CNom IN VARCHAR2,
            Credit IN INTEGER,
            Prerequis IN VARCHAR2)
      BEGIN
      INSERT INTO T_Cours_Type (
            CO_ID,
            CO_CNom,
            CO_Credit,
            CO_Prerequis
            )
      VALUES ( ID , CNom , Credit , Prerequis);
      END;

      MEMBER PROCEDURE CO_Delete
      BEGIN
      DELETE FROM T_Cours_Type WHERE CO_ID = SELF.CO_ID;
      END;
END;




CREATE OR REPLACE TYPE Etudiant_Cours_Type AS OBJECT (
      EC_Cours REF T_Cours_Type,
      EC_Etudiant REF T_Etudiant_Type
      );

CREATE TABLE Etudiant_Diplome OF Etudiant_Diplome_Type (
      Etudiant_EC SCOPE IS T_Etudiant,
      EC_Cours SCOPE IS T_Cours
      );




CREATE OR REPLACE TYPE T_Campus_Type AS OBJECT (
      CA_ID INTEGER,
      CA_SiteCampus VARCHAR2(100),
      CA_AdresseCampus VARCHAR2(300),
      CA_Telefone INTEGER,
      CA_Fax INTEGER,
      CA_Responsable VARCHAR2(100)
      T_PersonneS REF T_Personne_Type,
      T_BatimentS REF T_Batiment_Type,
      T_ResponsablePotentielS REF T_ResponsablePotentiel_Type
      );

CREATE TABLE T_Campus OF T_Campus_Type (
      PRIMARY KEY(CA_ID),
      T_PersonneS SCOPE IS T_Personne,
      T_BatimentS SCOPE IS T_Batiment
```

```
      T_ResponsablePotentielS SCOPE IS T_ResponsablePotentiel
      );

CREATE OR REPLACE TYPE T_Campus_Ref_Type AS OBJECT(
      T_Campus REF T_Campus_Type
      );
CREATE OR REPLACE TYPE T_CampusS_Type AS TABLE OF T_Campus_Ref_Type;

CREATE SEQUENCE CA_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER CA_Trigger BEFORE INSERT ON T_Campus
FOR EACH ROW
      BEGIN
            SELECT CA_Sequence.NEXTVAL INTO :NEW.CA_ID FROM DUAL;
      END;
```

```
CREATE OR REPLACE TYPE T_Admin_Type UNDER T_Personnel_Type (
      A_Titre VARCHAR2(100),
      A_CompInfo VARCHAR2(100),
      A_CompSecret VARCHAR2(100),
      MEMBER PROCEDURE A_Insert (
      Titre IN VARCHAR2,
      CompInfo IN VARCHAR2,
      CompSectret IN VARCHAR2,
      T_ResponsablePotentielS REF T_ResponsablePotentiel_Type
      ),
      MEMBER PROCEDURE A_Delete
      ) INSTANTIABLE, FINAL;

CREATE TABLE T_Admin OF T_Admin_Type (
      PRIMARY KEY(A_ID)
      )
      T_ResponsablePotentielS SCOPE IS T_ResponsablePotentiel;

CREATE SEQUENCE A_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER A_Trigger BEFORE INSERT ON T_Admin
FOR EACH ROW
      BEGIN
            SELECT A_Sequence.NEXTVAL INTO :NEW.A_ID FROM DUAL;
      END;

CREATE OR REPLACE BODY T_Admin_Type AS
      MEMBER PROCEDURE A_Insert (
      Titre IN VARCHAR2,
      CompInfo IN VARCHAR2,
      CompSecret IN VARCHAR2
      )
```

```
	BEGIN
	INSERT INTO T_Admin_Type(A_Titre, A_CompInfo, A_CompSecret) VALUES
(Titre, CompInfo , CompSectret);
	END;

	MEMBER PROCEDURE A_Delete
	BEGIN
	DELETE FROM T_Admin_Type WHERE CO_ID = SELF.CO_ID;
	END;
END;
```

```
CREATE OR REPLACE TYPE T_Professeur_Type AS OBJECT (
	PR_ID INTEGER,
	PR_Nom VARCHAR2(100),
	PR_Email VARCHAR2(100),
	T_Departement_Prof T_DepartementS_Type,
	T_CentreRecherche_Prof T_CentreRechercheS_Type,
	T_EcoleInstitut T_EcoleInstitutS_Type,
	T_ResponsablePotentielS  T_ResponsablePotentiel_Type
	);

CREATE TABLE T_Professeur OF T_Professeur_Type (
	PRIMARY KEY(PR_ID)
	)
	T_ResponsablePotentielS SCOPE IS T_ResponsablePotentiel,
	NESTED TABLE T_Departement_Prof  STORE AS PR_Departement,
	NESTED TABLE T_CentreRecherche STORE AS PR_CentreRecherche,
	NESTED TABLE T_EcoleInstitut1 PR_EcoleInstitut;

CREATE SEQUENCE PR_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER PR_Trigger BEFORE INSERT ON T_Professeur
FOR EACH ROW
	BEGIN
		SELECT PR_Sequence.NEXTVAL INTO :NEW.PR_ID FROM DUAL;
	END;
```

```
CREATE OR REPLACE TYPE T_Classe_Type AS OBJECT (
	CL_ID INTEGER,
	CL_Numero INTEGER,
	CL_Places INTEGER,
	T_Batiment_Classe T_BatimentS_Type
```

```
      );

CREATE TABLE T_Classe OF T_Classe_Type (
      PRIMARY KEY(CO_ID)
      )
      NESTED TABLE T_Batiment_Classe STORE AS CL_Batiment;

CREATE SEQUENCE CL_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER CL_Trigger BEFORE INSERT ON T_Classe
FOR EACH ROW
      BEGIN
            SELECT CL_Sequence.NEXTVAL INTO :NEW.CL_ID FROM DUAL;
      END;




CREATE OR REPLACE TYPE T_Lab_Tp_Type AS OBJECT (
      L_ID INTEGER,
      L_Numero INTEGER,
      L_Places INTEGER,
      L_Machine INTEGER,
      T_Batiment_L T_BatimentS_Type);

CREATE TABLE T_Lab_Tp OF T_Lab_Tp_Type (
      PRIMARY KEY(L_ID)
      )
      NESTED TABLE T_Batiment_L STORE AS L_Batiment;

CREATE SEQUENCE L_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER L_Trigger BEFORE INSERT ON T_Lab_Tp
FOR EACH ROW
      BEGIN
            SELECT L_Sequence.NEXTVAL INTO :NEW.L_ID FROM DUAL;
      END;




CREATE OR REPLACE TYPE T_Personne_Type AS OBJECT (
      P_ID INTEGER,
      P_Nom VARCHAR2(100),
      P_Prenom VARCHAR2(100),
      P_Titre VARCHAR2(100),
      P_Adresse VARCHAR2(300),
      P_Telefone INTEGER,
```

```
	P_CodePostal VARCHAR2(5),
	T_CampusS T_CampusS_Type,
	T_PersonnelS T_Personnel_Type,
	T_EtudiantS T_Etudiant_Type,
	MEMBER PROCEDURE P_Insert (
		ID IN INTEGER,
		Nom IN VARCHAR2,
		Prenom IN VARCHAR2,
		Titre IN VARCHAR2,
		Adresse IN VARCHAR2,
		Telefone IN VARCHAR2,
		CodePostal IN VARCHAR2
	),
	MEMBER PROCEDURE P_Delete); NOT FINAL INSTANTIABLE;

CREATE TABLE T_Personne OF T_Personne_Type (
	PRIMARY KEY(P_ID)
	)
	NESTED TABLE T_CampusS STORE AS CampusS_P,
	NESTED TABLE T_PersonnelS STORE AS PersonnelS_P,
	NESTED TABLE T_EtudiantS STORE AS EtudiantS_P;


CREATE SEQUENCE P_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER P_Trigger BEFORE INSERT ON T_Personne
FOR EACH ROW
	BEGIN
		SELECT P_Sequence.NEXTVAL INTO :NEW.P_ID FROM DUAL;
	END;

CREATE OR REPLACE BODY T_Personne_Type AS
	MEMBER PROCEDURE P_Insert (
	ID IN INTEGER,
	Nom IN VARCHAR2,
	Prenom IN VARCHAR2,
	Titre IN VARCHAR2,
	Adresse IN VARCHAR2,
	Telefone IN VARCHAR2,
	CodePostal IN VARCHAR2
	)
	BEGIN
	INSERT INTO T_Personne_Type ( P_ID , P_Nom, P_Prenom, P_Titre,
P_Adresse, P_Telefone, P_CodePostal )
	VALUES (ID , Nom , Prenom , Titre , Adresse , Telefone , CodePostal);
	END;

	MEMBER PROCEDURE P_Delete
	BEGIN
	DELETE FROM T_Personne_Type WHERE P_ID = SELF.P_ID;
	END;
```

```
CREATE OR REPLACE TYPE T_Technicien_Type UNDER T_Personnel_Type (
      T_ID INTEGER,
      T_Titre VARCHAR2(100),
      T_Competence VARCHAR2(100),
      MEMBER PROCEDURE T_Insert(ID IN INTEGER, TYP IN VARCHAR2, Titre IN
VARCHAR2, Competence IN VARCHAR2),
      MEMBER PROCEDURE T_Delete
      ) INSTANTIABLE, FINAL;

CREATE TABLE T_Technicien OF T_Technicien_Type(PRIMARY KEY(T_ID));

CREATE SEQUENCE T_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER T_Trigger BEFORE INSERT ON T_Technicien
FOR EACH ROW
      BEGIN
            SELECT T_Sequence.NEXTVAL INTO :NEW.T_ID FROM DUAL;
      END;

CREATE OR REPLACE BODY T_Technicien_Type AS MEMBER
      PROCEDURE T_Insert (ID IN INTEGER, TY IN VARCHAR2, Titre IN VARCHAR2,
Competence IN VARCHAR2)
      BEGIN
      INSERT INTO T_Technicien_Type(T_ID , PL_Type , T_Titre , T_Competence)
      VALUES (ID , TYP , Titre , Competence);
      END;


      MEMBER PROCEDURE T_Delete
      BEGIN
      DELETE FROM T_Technicien_Type WHERE T_ID = SELF.T_ID AND WHERE PL_Type
= SELF.PL_Type AND T_Titre = SELF.T_Titre;
      END;
END;



CREATE OR REPLACE TYPE T_ChargeCours_Type UNDER T_Personnel_Type (
      CC_ID INTEGER,
      CC_Domaine VARCHAR2(100),
      CC_TypeCours VARCHAR2(100),
      T_Cours_CC REF T_Cours_Type,
      MEMBER PROCEDURE afficheCharge RETURN VARCHAR2
      ) NOT INSTANTIABLE, NOT FINAL;

CREATE TABLE T_ChargeCours OF T_ChargeCours_Type(PRIMARY KEY(CC_ID),
T_Cours_CC SCOPE IS T_Cours);

CREATE SEQUENCE CC_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER CC_Trigger BEFORE INSERT ON T_ChargeCours
```

```
FOR EACH ROW
      BEGIN
            SELECT CC_Sequence.NEXTVAL INTO :NEW.CC_ID FROM DUAL;
      END;

CREATE OR REPLACE BODY T_ChargeCours AS
      MEMBER FUNCTION afficheCharge RETURN VARCHAR2 IS
      resultat VARCHAR2(6000);
      BEGIN
      SELECT CC_ID||' '||CC_Domaine||' '||PL_Type||' '||CC_TypeCours INTO
resultat FROM T_ChargeCours CC
      WHERE CC.CC_ID = SELF.CC_ID AND CC_TypeCours = SELF.CC_TypeCours AND
CC.PL_Type = SELF.PL_Type;
      RETURN resultat;
      END;
END;
```

```
CREATE OR REPLACE TYPE T_Tuteur_Type UNDER T_Personnel_Type (
      TU_ID INTEGER,
      TU_Heure INTEGER,
      TU_TauxHeures VARCHAR2(100),
      MEMBER PROCEDURE TY_Insert ( ID IN VARCHAR2 , TYP IN VARCHAR2 , Heure
IN VARCHAR2 , TauxHeures IN VARCHAR2 ),
      MEMBER PROCEDURE TU_Delete
) INSTANTIABLE, FINAL;

CREATE TABLE T_Tuteur OF T_Tuteur_Type(PRIMARY KEY(T_ID));

CREATE SEQUENCE TU_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER TU_Trigger BEFORE INSERT ON T_Tuteur
FOR EACH ROW
      BEGIN
            SELECT TU_Sequence.NEXTVAL INTO :NEW.TU_ID FROM DUAL;
      END;

CREATE OR REPLACE BODY T_Tuteur_Type AS
      MEMBER PROCEDURE TU_Insert ( ID IN VARCHAR2 , TYP IN VARCHAR2 , Heure
IN VARCHAR2 , TauxHeures IN VARCHAR2 )
      BEGIN
      INSERT INTO TU_Tuteur_Type(TU_ID , PL_Type , TU_Heure , TU_TauxHeures )
VALUES ( ID,  TYP , Heure, TauxHeures);
      END;

      MEMBER PROCEDURE TU_Delete
      BEGIN
      DELETE FROM T_Tuteur_Type WHERE PL_Type = SELF.PL_Type AND TU_Heure =
SELF.TU_Heure AND TU_TauxHeures = SELF.TU_TauxHeures;
```

```
	END;
END;



CREATE OR REPLACE TYPE T_Enseignant_Associate_Type UNDER T_ChargeCours_Type (
	EA_ID INTEGER,
	EA_Doctorat VARCHAR2(100),
	EA_Maitrise VARCHAR2(100),
	MEMBER PROCEDURE EA_Insert,
	MEMBER PROCEDURE EA_Delete) INSTANTIABLE, FINAL;

CREATE TABLE T_Enseignant_Associate OF T_Enseignant_Associate_Type (
	PRIMARY KEY(EA_ID)
	);

CREATE SEQUENCE EA_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER EA_Trigger BEFORE INSERT ON
T_Enseignant_Associate_Type
FOR EACH ROW
	BEGIN
		SELECT EA_Sequence.NEXTVAL INTO :NEW.EA_ID FROM DUAL;
	END;

CREATE OR REPLACE BODY T_Enseignant_Associate_Type AS
	MEMBER PROCEDURE EA_Insert ( ID INTEGER , TYP IN VARCHAR2 , Doctorat IN
VARCHAR2, Maitrise IN VARCHAR2)
	BEGIN
	INSERT INTO T_Enseignant_Associate_Type( EA_ID , PL_Type , EA_Doctorat
, EA_Maitrise )
	VALUES ( ID , TYP , Doctorat , Maitrise);
	END;

	MEMBER PROCEDURE EA_Delete
	BEGIN
	DELETE FROM T_Enseignant_Associate_Type WHERE PL_Type = SELF.PL_Type
AND EA_Doctorat = SELF.EA_Doctorat AND EA_Maitrise = SELF.EA_Maitrise;
	END;
END;



CREATE OR REPLACE TYPE T_Enseignant_Vacataire_Type UNDER T_ChargeCours_Type (
	EV_ID INTEGER,
	EV_Diplome VARCHAR2(100),
	EV_AnneeRecrute INTEGER,
	MEMBER PROCEDURE EV_Insert
	MEMBER PROCEDURE EV_Delete
	) INSTANTIABLE, FINAL;
```

```
CREATE TABLE T_Enseignant_Vacataire OF T_Enseignant_Vacataire_Type (
      PRIMARY KEY(EV_ID)
      );

CREATE SEQUENCE EV_Sequence START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER EV_Trigger BEFORE INSERT ON
T_Enseignant_Associate_Type
FOR EACH ROW
      BEGIN
            SELECT EV_Sequence.NEXTVAL INTO :NEW.EV_ID FROM DUAL;
      END;

CREATE OR REPLACE BODY T_Enseignant_Vacataire_Type AS
      MEMBER PROCEDURE EV_Insert (ID IN VARCHAR2 , TYP IN VARCHAR2, Diplome
IN VARCHAR2, AnneeRecrute IN VARCHAR2)
      BEGIN
      INSERT INTO T_Enseignant_Vacataire_Type( EV_ID INTEGER , PL_Type ,
EV_Diplome , EV_AnneeRecrute ) VALUES ( ID , TYP , Diplome , AnneeRecrute );
      END;

      MEMBER PROCEDURE EV_Delete
      BEGIN
      DELETE FROM T_Enseignant_Vacataire_Type WHERE PL_Type = SELF.PL_Type
AND EV_Diplome = SELF.EV_Diplome AND EV_AnneeRecrute = SELF.EV_AnneeRecrute;
      END;



CREATE OR REPLACE TYPE T_ResponsablePotentiel_Type AS OBJECT (
      PR_ID INTEGER,
      A_ID INTEGER,
      CA_ID INTEGER,
      T_ProfesseurS T_Professeur_Type,
      T_AdminS T_Admin_Type,
      T_CampusS T_Campus_Type,
      );

CREATE TABLE T_ResponsablePotentiel OF T_ResponsablePotentiel_Type (
      PRIMARY KEY(RE_ID)
      )
      NESTED TABLE T_ProfesseurS STORE AS RE_Professeur,
      NESTED TABLE T_AdminS STORE AS RE_Admin,
      NESTED TABLE T_CamousS STORE AS RE_Campus;

CREATE OR REPLACE TYPE T_ResponsablePotentiel_Ref_Type AS OBJECT (
      T_ResponsablePotentiel REF T_ResponsablePotentiel_Type
      );
CREATE OR REPLACE TYPE T_ResponsablePotentielS_Type AS TABLE OF
T_ResponsablePotentiel_Ref_Type;


CREATE SEQUENCE RE_Sequence START WITH 1 INCREMENT BY 1;
```

```
CREATE OR REPLACE TRIGGER RE_Trigger BEFORE INSERT ON
T_ResponsablePotentiel_Type
FOR EACH ROW
      BEGIN
            SELECT RE_Sequence.NEXTVAL INTO :NEW.RE_ID FROM DUAL;
      END;
```