UNIVERSITE PARIS DAUPHINE

2003-2004

Université Paris Dauphine
IUP Génie Mathématique et Informatique
IUP 2ème année et Formation Continue

BASES DE DONNEES

TP POSTGRESQL

Maude Manouvrier

2003-2004

Université Paris Dauphine
IUP MIAGE
IUP 2ème année et Formation Continue

BASES DE DONNEES

TP POSTGRESQL

Maude Manouvrier

TABLE DES MATIERES

PostgreSQL est un Système de Gestion de Bases de Données Relationnel Objet [GS02], *open source*, prédécesseur de Ingres, développé par l'Université de Californie de Berkeley. Pour plus d'informations sur PostgreSQL, vous pouvez regarder les sites suivants :
http://www.postgresql.org
http://www.grappa.univ-lille3.fr/polys/reseaux-2000/reseaux022.html
http://www.commandprompt.com/ppbook/index.lxp?lxpwrap=book1.htm

Ce document a pour objectif de vous aider à utiliser ce SGBD. Il contient le script de la base de données exemple (voir Section I) et en annexe les scripts des nuplets insérés(voir en Annexe – Section V), des exemples de fonctions SQL, PL/pgSQL et de déclencheur (voir Section II), une description de l'interface de l'outil PgAdmin (voir Section III), un exemple de programme ODBC (voir Section IV), ainsi que le sujet du TP (voir Section IV). L'annexe contient également un petit guide d'installation de PosgreSQL sous Windows (voir Section VI).

I. SCRIPT DE LA BASE DE DONNEES EXEMPLE

Les commandes `DROP` ne sont à utiliser que lorsque les relations existent déjà et que l'on souhaite les supprimer. *Lors de l'initialisation de la base exemple, il faut supprimer ces commandes.*

```
DROP VIEW Email_Etudiant;
DROP TABLE Etudiant;
DROP TABLE Reservation;
DROP TABLE Salle;
DROP TABLE Enseignement;
DROP TABLE Enseignant;
DROP TABLE Departement;


CREATE TABLE Departement
(
 Departement_id    integer,
 Nom_Departement   varchar(25) NOT NULL,
 CONSTRAINT UN_Nom_Departement UNIQUE (nom_departement),
 CONSTRAINT PK_Departement PRIMARY KEY(Departement_ID)
);

CREATE TABLE Etudiant
(
 Etudiant_ID     integer,
 Nom             varchar(25) NOT NULL,
 Prenom          varchar(25) NOT NULL,
 Date_Naissance  date NOT NULL,
 Adresse         varchar(50) DEFAULT NULL,
 Ville           varchar(25) DEFAULT NULL,
 Code_Postal     varchar(9) DEFAULT NULL,
 Telephone       varchar(10) DEFAULT NULL,
 Fax             varchar(10) DEFAULT NULL,
 Email           varchar(100) DEFAULT NULL,
 CONSTRAINT PK_Etudiant PRIMARY KEY (Etudiant_ID)
);
```

```
CREATE TABLE Enseignement
(
  Enseignement_ID  int4 NOT NULL,
  Departement_ID   int4 NOT NULL,
  Intitule         varchar(60) NOT NULL,
  Description      varchar(1000),
  CONSTRAINT PK_Enseignement
    PRIMARY KEY (Enseignement_ID, Departement_ID),
  CONSTRAINT "PK_Enseignement_Departement"
    FOREIGN KEY (Departement_ID)
     REFERENCES Departement (Departement_ID)
      ON UPDATE RESTRICT ON DELETE RESTRICT
);


CREATE TABLE Enseignant
(
 Enseignant_ID     integer,
 Departement_ID    integer NOT NULL,
 Nom               varchar(25) NOT NULL,
 Prenom            varchar(25) NOT NULL,
 Grade             varchar(25)
 CONSTRAINT CK_Enseignant_Grade
 CHECK (Grade IN ('Vacataire', 'Moniteur','ATER', 'MCF', 'PROF')),
 Telephone         varchar(10) DEFAULT NULL,
 Fax               varchar(10) DEFAULT NULL,
 Email             varchar(100) DEFAULT NULL,
 CONSTRAINT PK_Enseignant PRIMARY KEY (Enseignant_ID),
 CONSTRAINT "FK_Enseignant_Departement_ID"
   FOREIGN KEY (Departement_ID)
    REFERENCES Departement (Departement_ID)
      ON UPDATE RESTRICT ON DELETE RESTRICT
);


CREATE TABLE Salle
(
 Batiment       varchar(1),
 Numero_Salle   varchar(10),
 Capacite       integer CHECK (Capacite >1),
 CONSTRAINT PK_Salle PRIMARY KEY (Batiment, Numero_Salle)
);
```

```
CREATE TABLE Reservation
(
 Reservation_ID     integer,
 Batiment           varchar(1) NOT NULL,
 Numero_Salle       varchar(10) NOT NULL,
 Enseignement_ID    integer NOT NULL,
 Departement_ID     integer NOT NULL,
 Enseignant_ID      integer NOT NULL,
 Date_Resa          date NOT NULL DEFAULT CURRENT_DATE,
 Heure_Debut        time NOT NULL DEFAULT CURRENT_TIME,
 Heure_Fin          time NOT NULL DEFAULT '23:00:00',
 Nombre_Heures      integer NOT NULL,
 CONSTRAINT PK_Reservation PRIMARY KEY (Reservation_ID),
 CONSTRAINT "FK_Reservation_Salle"
   FOREIGN KEY (Batiment,Numero_Salle)
    REFERENCES Salle (Batiment,Numero_Salle)
     ON UPDATE RESTRICT ON DELETE RESTRICT,
 CONSTRAINT "FK_Reservation_Enseignement"
   FOREIGN KEY (Enseignement_ID,Departement_ID)
    REFERENCES Enseignement (Enseignement_ID,Departement_ID)
     ON UPDATE RESTRICT ON DELETE RESTRICT,
 CONSTRAINT "FK_Reservation_Enseignant"
   FOREIGN KEY (Enseignant_ID)
    REFERENCES Enseignant (Enseignant_ID)
     ON UPDATE RESTRICT ON DELETE RESTRICT,
 CONSTRAINT CK_Reservation_Nombre_Heures CHECK (Nombre_Heures >=1),
 CONSTRAINT CK_Reservation_HeureDebFin
   CHECK (Heure_Debut < Heure_Fin)
);

CREATE OR REPLACE VIEW Email_Etudiant
 AS SELECT Nom, Prenom, Email FROM Etudiant;
```

II. EXEMPLE DE FONCTIONS SQL, PL/PGSQL ET DE TRIGGER

A. Fonctions SQL

```
CREATE OR REPLACE FUNCTION GetSalleCapaciteSuperieurA(int)
  RETURNS SETOF Salle
   AS '
        SELECT * FROM Salle WHERE Capacite > $1;
      '
LANGUAGE SQL;
```

La fonction ci-dessus prend en paramètre un entier correspondant à la capacité voulue pour une salle et retourne un ensemble de nuplets de la relation *Salle* ayant une capacité supérieure au paramètre. Le paramètre est représenté par $1 dans le corps de la fonction.

La requête ci-dessous permet par exemple d'appeler cette fonction pour rechercher les salles de capacité supérieure à 300.

```
SELECT * FROM GetSalleCapaciteSuperieurA(300) ;
```

```
CREATE OR REPLACE FUNCTION GetDepartement_ID(text) RETURNS integer AS
'SELECT Departement_ID FROM Departement WHERE Nom_Departement = $1'
LANGUAGE SQL;
```

La fonction ci-dessus prend en paramètre un nom de département et retourne l'identificateur du département correspondant.

La requête ci-dessous permet par exemple d'appeler cette fonction pour rechercher le département 'INFO'.

```
SELECT Nom, Prenom
 FROM Enseignant
  WHERE Departement_ID IN (SELECT * FROM GetDepartement_ID('INFO'));
```

```
CREATE OR REPLACE FUNCTION PossibiliteResa(text,text,date,time,time)
RETURNS integer AS
'SELECT Reservation_ID
  FROM Reservation
   WHERE (Batiment = $1 AND Numero_Salle = $2 AND Date_Resa = $3
     AND Heure_Debut < $4  AND $4 < Heure_Fin)
   OR (Batiment = $1 AND Numero_Salle = $2 AND Date_Resa = $3
    AND $4 < Heure_Debut AND Heure_Debut < $5 AND Heure_Fin > $5)
   OR (Batiment = $1 AND Numero_Salle = $2 AND Date_Resa = $3
    AND Heure_Debut < $4 AND $4 < Heure_Fin AND Heure_Fin < $5)
   OR (Batiment = $1 AND Numero_Salle = $2 AND Date_Resa = $3
    AND Heure_Debut = $4 AND Heure_Fin = $5)
   OR (Batiment = $1 AND Numero_Salle = $2 AND Date_Resa = $3
    AND Heure_Debut = $4)
   OR (Batiment = $1 AND Numero_Salle = $2 AND Date_Resa = $3
    AND Heure_Fin = $5)'
LANGUAGE SQL;
```

La fonction ci-dessus vérifie que le créneau horaire choisit pour une réservation n'est pas contenu dans le(s) créneau(x) horaire(s) de réservations existantes ou ne chevauche pas le(s) créneau(x) horaire(s) de réservations existantes. Elle prend en paramètre un numéro de bâtiment et un numéro de salle (sous forme de chaînes de caractères), une date de réservation et une heure de début et de fin de réservation et retourne les identificateurs des réservations qui rendent la réservation demandée impossible (ou ne retourne rien sinon). Dans le corps de la fonction, le bâtiment est représenté par $1, le numéro de salle par $2, la date de réservation par $3, l'heure de début par $4 et l'heure de fin de réservation par $5.

La requête ci-dessous permet par exemple d'appeler cette fonction pour voir s'il est possible de réserver la salle B022 le 4 novembre 2003 entre 9h et 18h.

```
SELECT PossibiliteResa('B','022','04/11/2003','09:00:00','18:00:00');
```

B. Fonction trigger en PL/pgSQL

1. Activation du PL/pgSQL

Le langage PL/pgSQL est un langage procédural (équivalent au PL/SQL sous Oracle) permettant d'intégrer des commandes SQL, avec des déclarations de variables, des boucles, etc. Pour activer le langage PL/pgSQL il faut dans `cygwin`[1] exécuter la commande :

`createlang plpgsql nom_base`

2. Exemple de fonction utilisée dans un déclencheur

```
CREATE OR REPLACE FUNCTION FunctionTriggerReservation() RETURNS trigger AS
' DECLARE
  resa Reservation.Reservation_ID%TYPE;
  BEGIN
   SELECT INTO resa Reservation_ID
    FROM Reservation
     WHERE (Batiment = NEW.Batiment AND Numero_Salle = NEW.Numero_Salle
        AND Date_Resa = NEW.Date_Resa AND Heure_Debut < NEW.Heure_Debut
         AND NEW.Heure_Debut < Heure_Fin)
      OR (Batiment = NEW.Batiment AND Numero_Salle = NEW.Numero_Salle
        AND Date_Resa = NEW.Date_Resa AND NEW.Heure_Debut < Heure_Debut
         AND Heure_Debut < NEW.Heure_Fin AND Heure_Fin > NEW.Heure_Fin)
      OR (Batiment = NEW.Batiment AND Numero_Salle = NEW.Numero_Salle
       AND Date_Resa = NEW.Date_Resa AND Heure_Debut < NEW.Heure_Debut
         AND NEW.Heure_Debut < Heure_Fin AND Heure_Fin < NEW.Heure_Fin)
      OR(Batiment = NEW.Batiment AND Numero_Salle = NEW.Numero_Salle
        AND Date_Resa = NEW.Date_Resa AND Heure_Debut = NEW.Heure_Debut
          AND Heure_Fin = NEW.Heure_Fin)
      OR (Batiment = NEW.Batiment AND Numero_Salle = NEW.Numero_Salle
       AND Date_Resa = NEW.Date_Resa AND Heure_Debut = NEW.Heure_Debut)
      OR (Batiment = NEW.Batiment AND Numero_Salle = NEW.Numero_Salle
        AND Date_Resa = NEW.Date_Resa AND Heure_Fin = NEW.Heure_Fin);
  IF FOUND THEN
    RAISE EXCEPTION ''Réservation impossible, salle occupée à la date
    et aux horaires demandés'';
  ELSE RETURN NEW;
  END IF;
END;'
LANGUAGE 'plpgsql';
```

← *Déclaration d'une variable qui va recevoir les valeurs des attributs Reservation_ID retournés par la requête. %TYPE permet de préciser le type de la variable (elle a pour type celui de l'attribut)*

← *Si on peut faire l'insertion, la fonction retourne le nuplet en cours d'insertion, représenté par NEW.*

Cette fonction ci-dessus est utilisée dans un déclencheur (ou *trigger* en anglais – vois section suivante). Lors d'une insertion d'une réservation dans la base de données (le nuplet inséré étant représenté par la variable **`NEW`**), elle va vérifier que cette réservation est possible (reprise du code de la fonction SQL `PossibiliteResa` expliquée précédemment) et si ce n'est pas le cas, va afficher un message d'erreur. Si l'insertion est possible, le nuplet à insérer est retourné.

C. Trigger

```
CREATE TRIGGER InsertionReservation
 BEFORE INSERT ON Reservation
  FOR EACH ROW
   EXECUTE PROCEDURE FunctionTriggerReservation();
```

Le déclencheur ci-dessus s'exécute avant l'insertion de tout nuplet dans la table *Réservation*. Il fait appel à la fonction `InsertionReservation`, expliquée précédemment.

[1] `Cygwin` est un interpréteur de commandes à la UNIX pour Windows. Voir à l'adresse http://www.cygwin.com/

III. INTERFACE DE L'OUTIL PGADMIN

Cette section a été rédigée en collaboration avec A. Bahri et Y. Naïja.

`PgAdmin` est un outil graphique permettant de manipuler PostgreSQL. Pour plus d'informations, vous pouvez consulter l'adresse : http://pgadmin.postgresql.org/pgadmin3/

A. Lancer PgAdmin

Pour accéder au logiciel depuis le CRIO UNIX, sous NT, aller menu **Démarrer**, le sous-menu **Programmes**, le sous menu **pgAdmin III**.

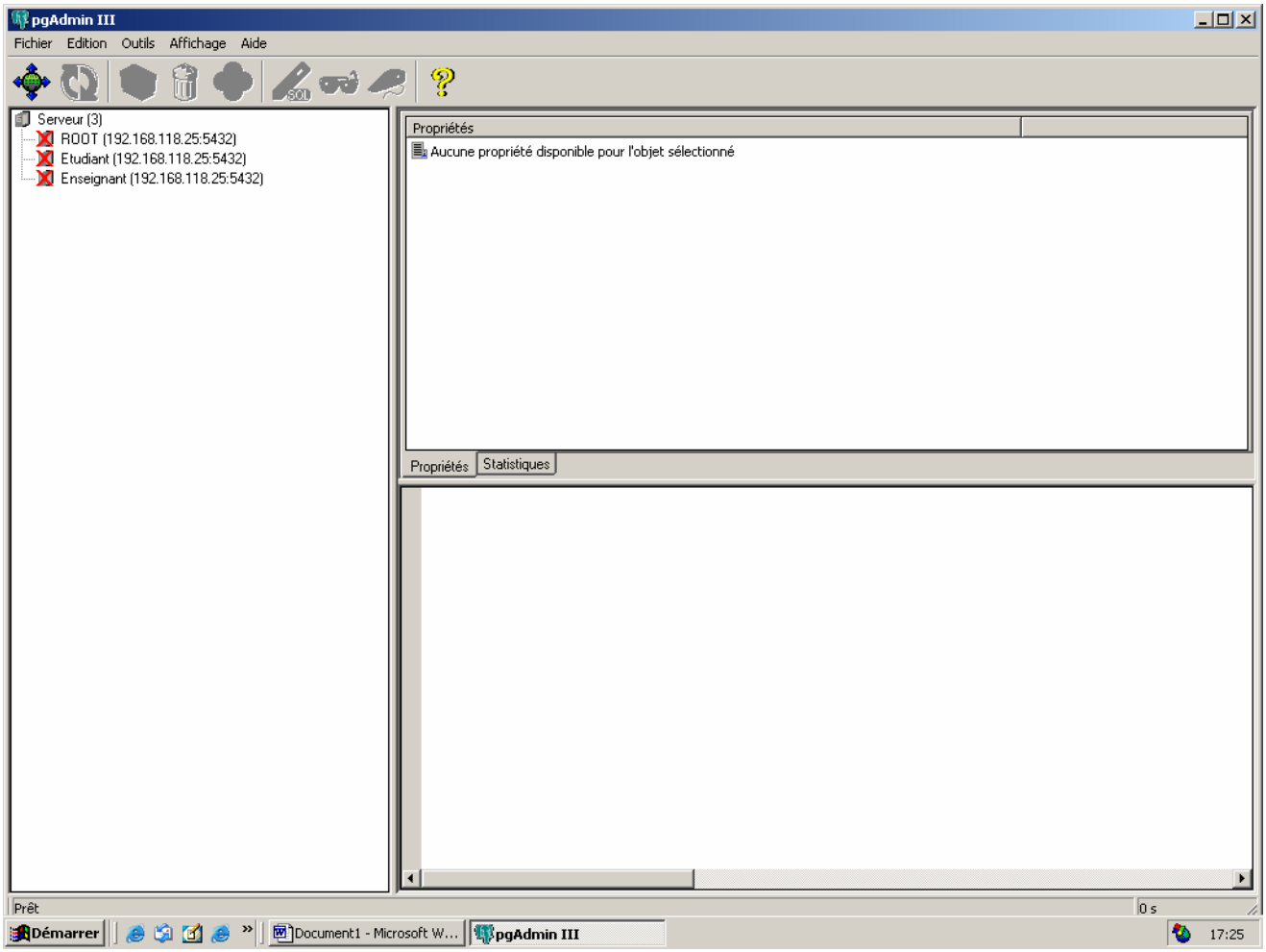

Figure 1 - Fenêtre principale de PgAdmin

B. Se connecter

Pour vous connecter, cliquez avec le bouton droit sur votre nom d'utilisateur, choisissez **Connexion** (ou double-cliquez sur votre nom utilisateur)**.** Une fenêtre apparaît. Dans cette fenêtre, tapez le numéro IP du serveur (**192.168.118.202**), votre login et votre mot de passe et cliquez sur OK.

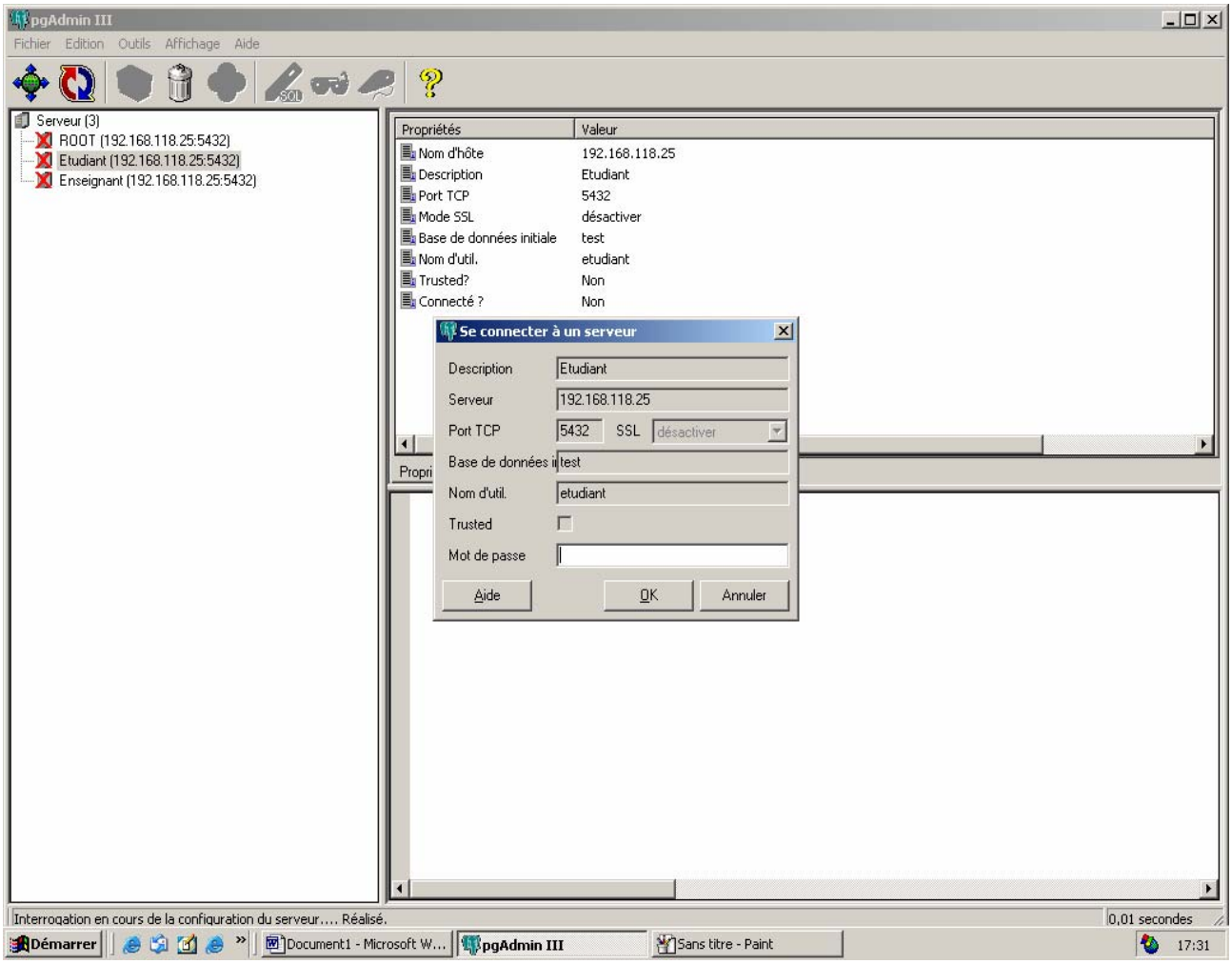

Figure 2 – Connexion à un serveur pour une base de données.

C. Exécuter des commandes SQL

Pour exécuter des commandes SQL, cliquez sur le bouton **SQL** de la barre à outils. Une fenêtre apparaît.

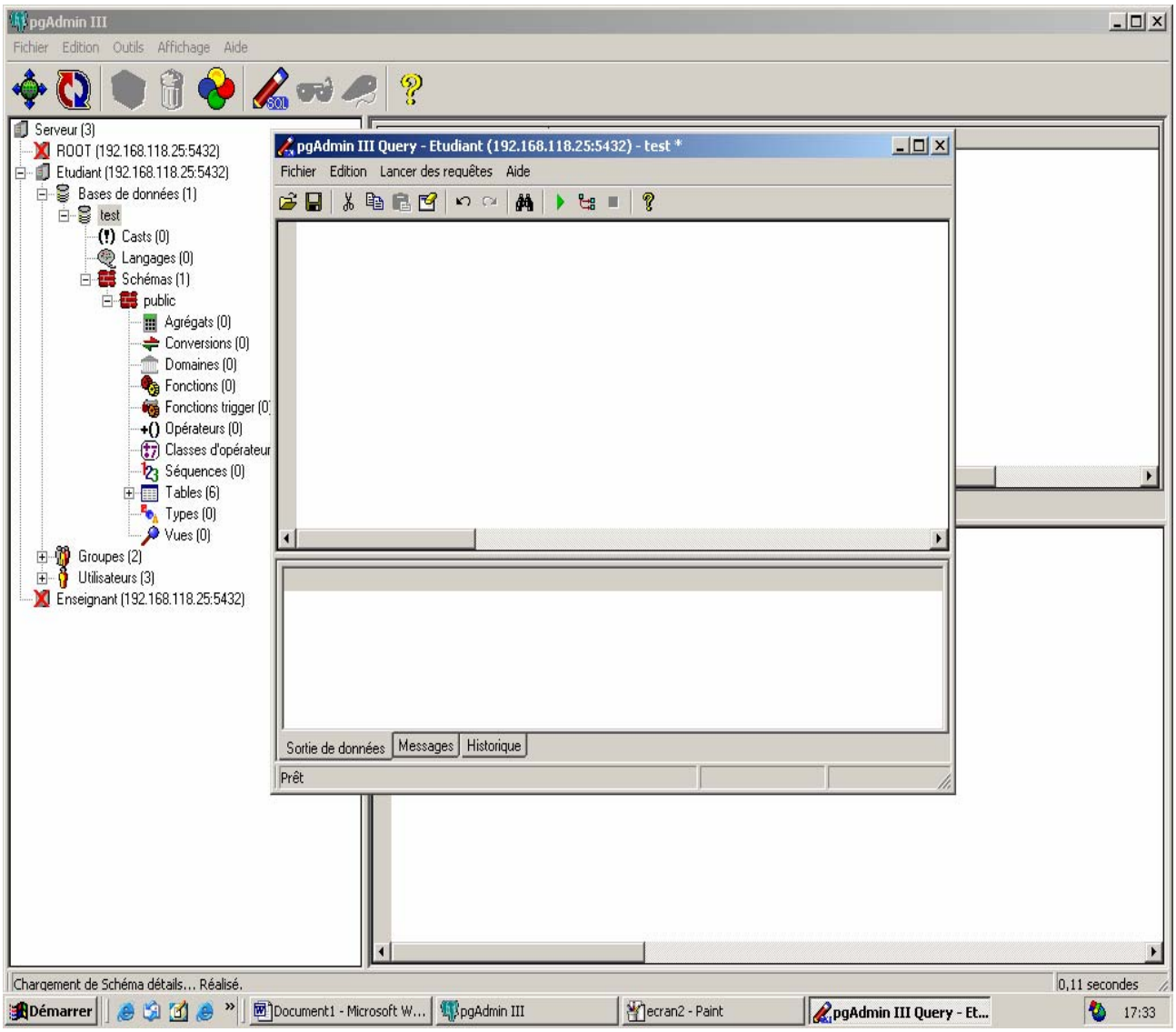

Figure 3 - Interpréteur de commandes SQL de PgAdmin.

La requête SQL doit être rédigée dans la fenêtre du haut. Pour exécuter la requête, cliquez sur l'icône en vert de la barre d'outils ou allez dans le menu **Lancer des requêtes**.

Les commentaires dans une requête commencent par --

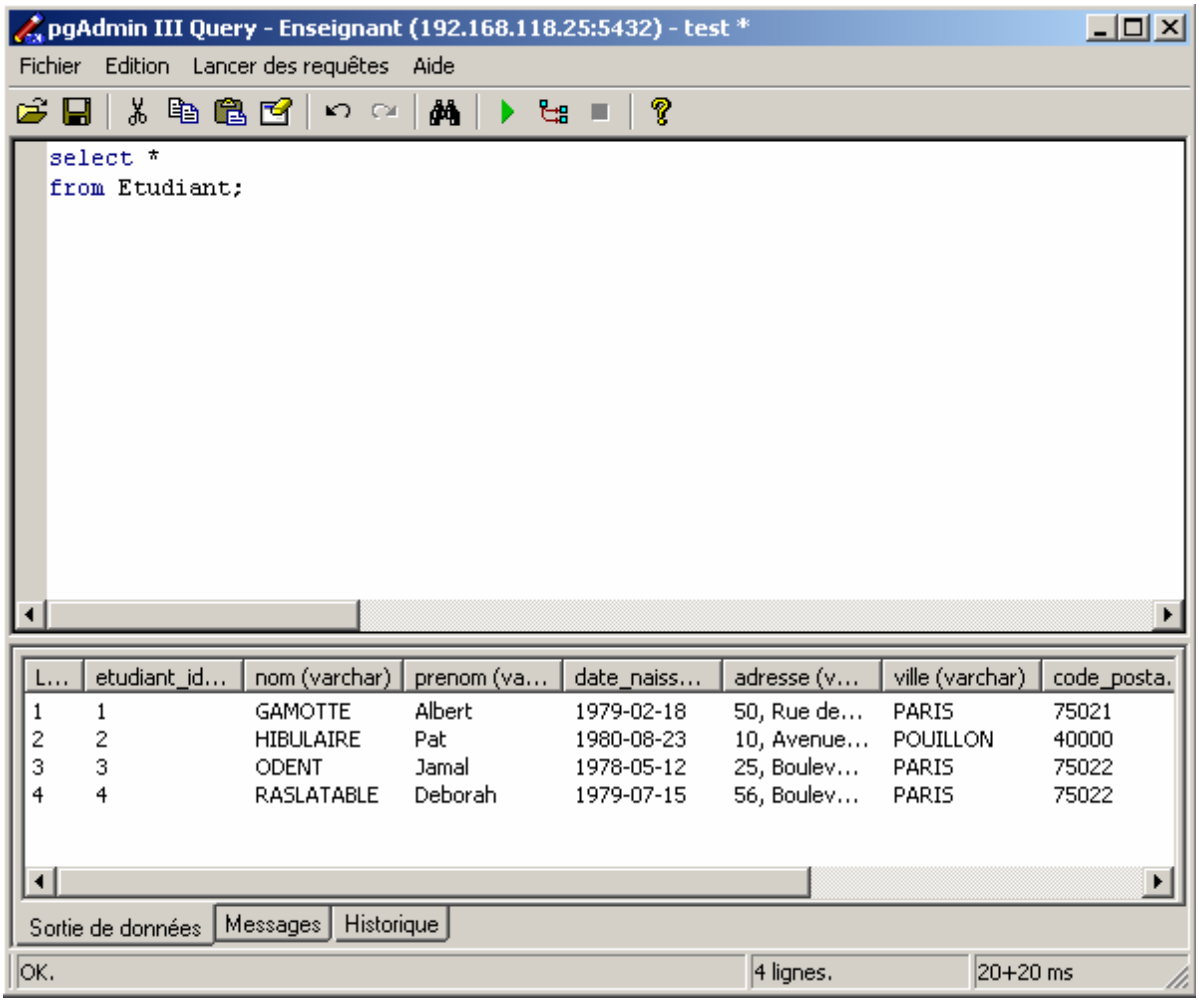

Figure 4 - Exemple de requête exécutée dans l'interpréteur de commandes SQL.

D. Visualiser les relations de la base de données

Les relations de la base de données sont accessibles en cliquant sur Tables à gauche de la fenêtre principale de pgAdmin. Lorsque vous cliquez sur le nom d'une table, son schéma au format SQL apparaît en bas à droite de la fenêtre.

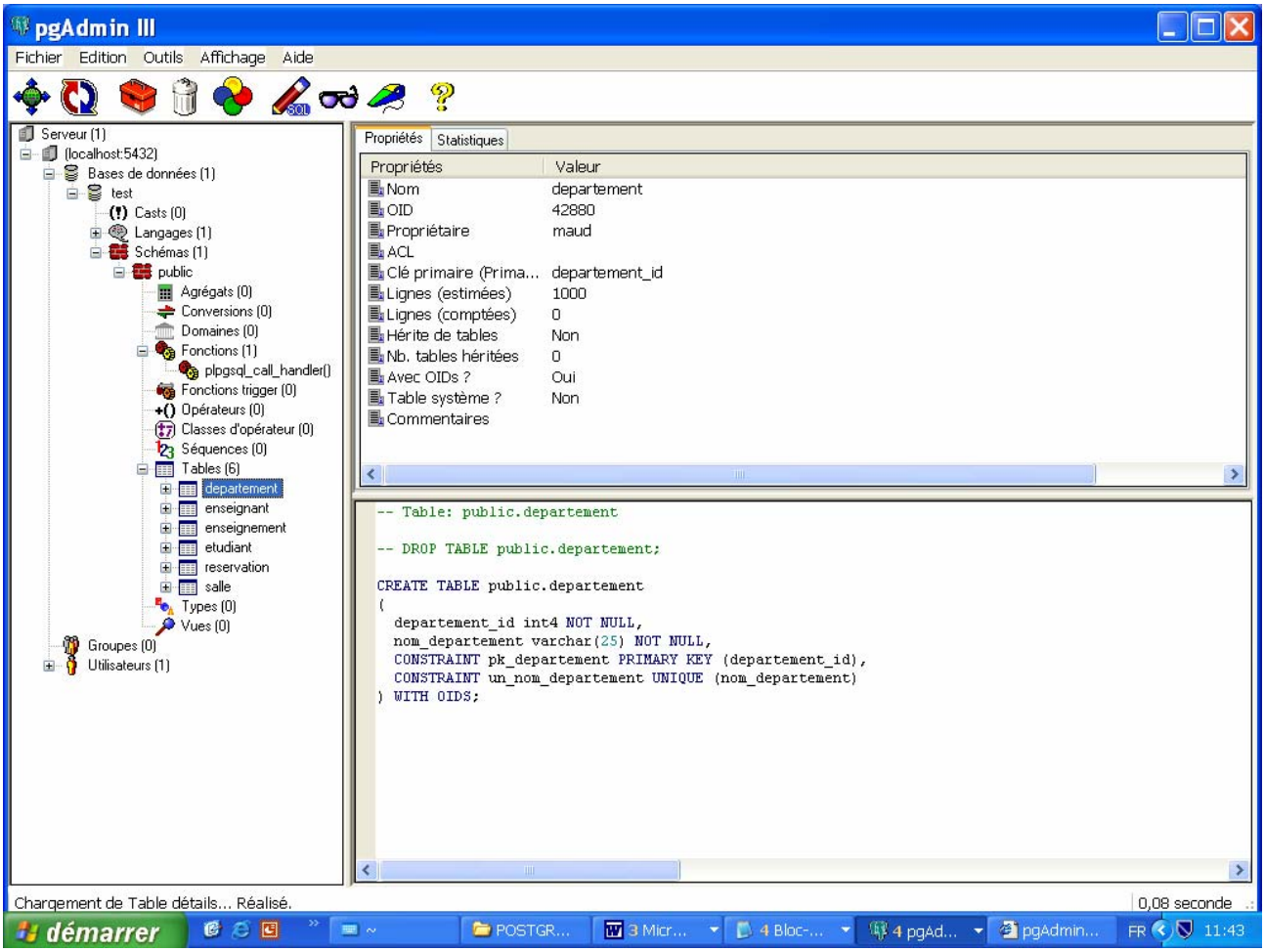

Figure 5 - Visualisation du script SQL de création de la relation *Département*.

En cliquant sur le bouton droit de la souris, vous pouvez accéder aux propriétés de la relation et en particulier visualiser les nuplets contenus.

E. Accès à l'aide

Vous pouvez accéder à l'aide de PgAdmin (en particulier à l'aide des commandes SQL) via le menu ou en appuyant sur la touche *F1*.

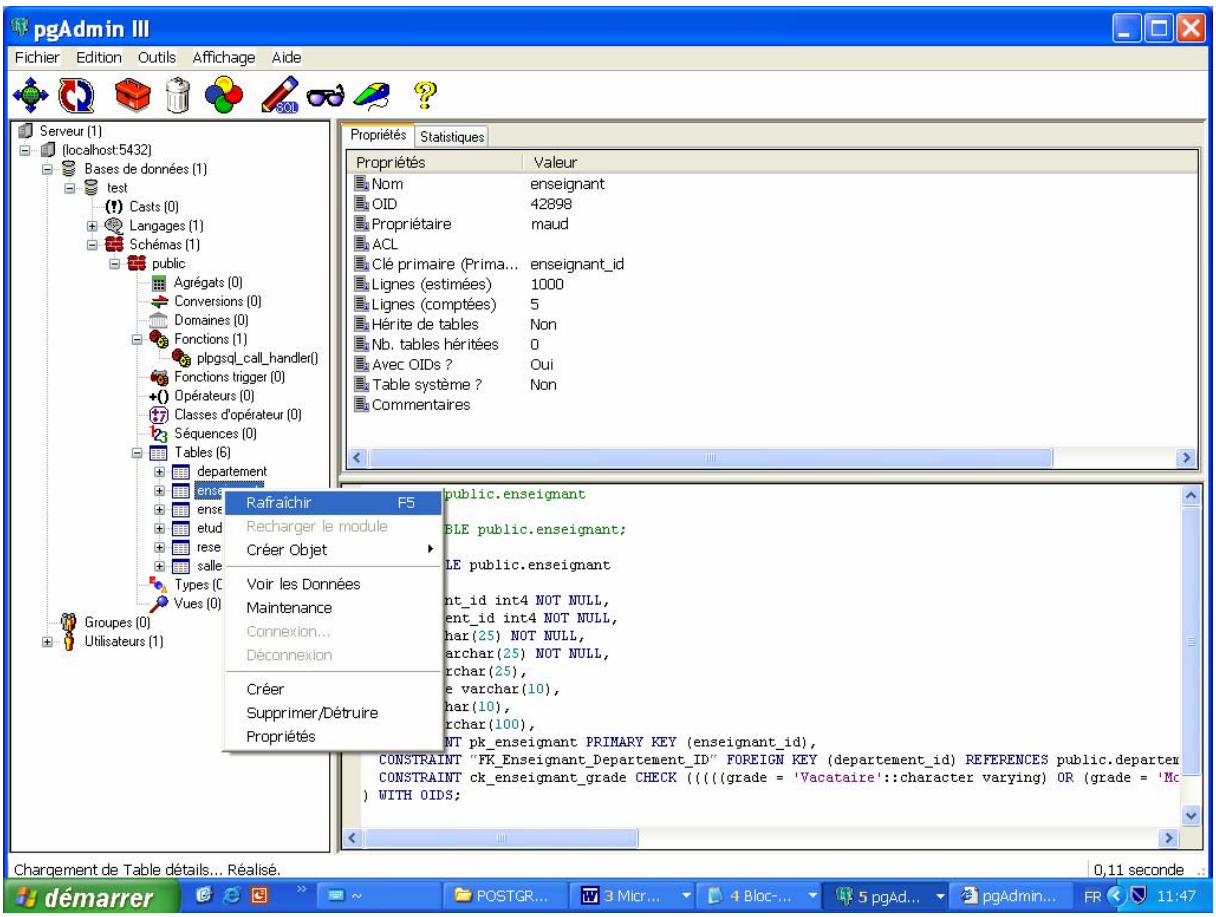

Figure 6 - Accès aux propriétés d'une relation (en particulier accès aux données).

IV. MANIPULATION D'UNE SOURCE DE DONNEES VIA ODBC

ODBC est un *middelware* facilitant la connexion entre un client de bases de données **sous Windows**[2] et un serveur de base de données. La connexion se fait via le **gestionnaire ODBC**. La manipulation se fait par une interface de programmation (*Application Programming Interface - API*) qui permet au programmeur d'accéder aux bases de données de manière transparente, c'est-à-dire indépendamment du SGBD utilisé. Un même programme, via l'**API ODBC**, peut interroger différentes bases de données sur différentes plates-formes. Le langage de l'API d'ODBC est une combinaison d'appels systèmes et de SQL. Il existe pour chaque SGBD, un **pilote ODBC** (ou *driver*) particulier. Ce pilote permet la traduction des commandes ODBC en commandes spécifiques au SGBD utilisé.

A. Connexion d'une source de données à ODBC

Une source de données est un nom logique de bases de données pour ODBC**. Pour connecter une source de données à ODBC** : sélectionnez, dans le menu **Démarrer**, sous-menu **Paramètres**, le sous-menu **Panneau de Configuration**. Puis, double-cliquez sur **Outils d'administration** puis sur l'icône ODBC

[2] Il existe des implantation d'ODBC sous d'autres plates-formes.

Figure 7 - Panneau de configuration pour accéder aux services ODBC.

Une fenêtre apparaît, contenant plusieurs volets :

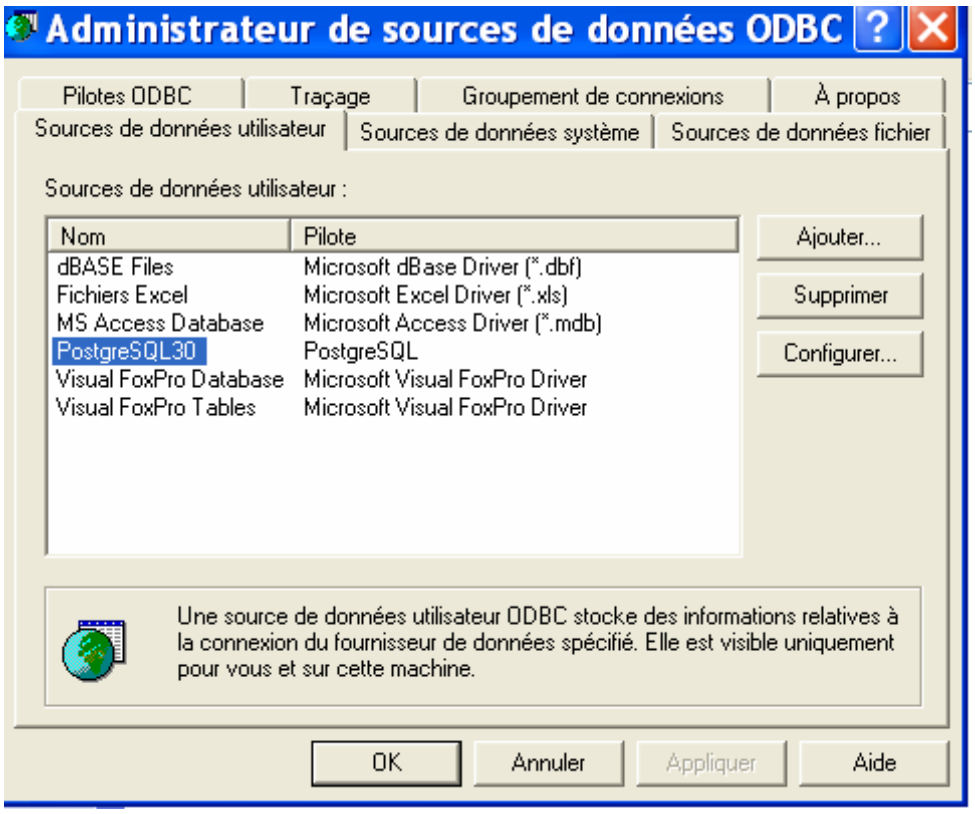

Figure 8 - Choix d'une source de données ODBC.

Si dans le volet **DNS Utilisateur** ou **Sources de données utilisateur**, (pour une connexion de l'utilisateur connecté sur la machine) ou dans le volet **DNS Système** (pour une connexion relative à la machine, quel que soit l'utilisateur connecté), le nom du pilote PostgreSQL apparaît, cliquez sur OK pour quitter la fenêtre.

Une fenêtre apparaît où vous devez saisir le nom de votre base de données, le serveur (adresse IP associée) et votre login sous PostgreSQL :

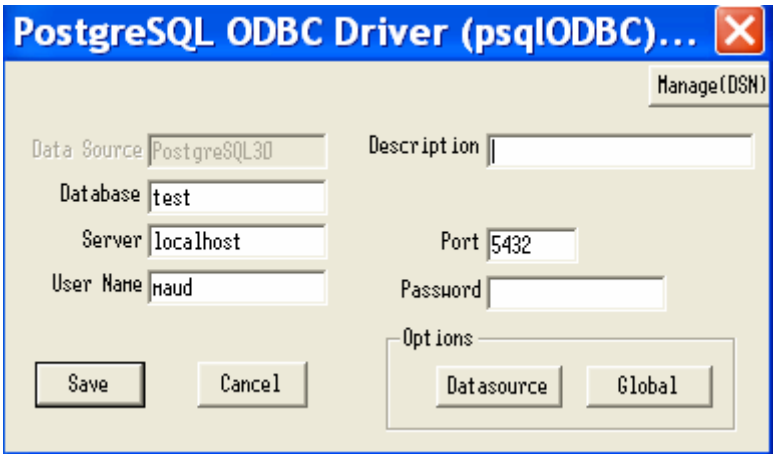

Figure 9 - Fenêtre de source de données ODBC propre à PosgreSQL.

Par défaut, la source de données PostgreSQL s'appelle PostgreSQL30.

B. Visual C++

Sur les machines NT, vous avez à votre disposition l'environnement de développement Visual C++ 6.0. Ce logiciel est accessible dans le menu **Démarrer**, sous-menu **Programmes**, le sous-menu **Microsoft Visual C++ 6.0.**

1. Création d'un programme

Pour créer un programme C sous Microsoft Visual C++ 6.0, dans le menu **File**, cliquez sur **New**. Une fenêtre apparaît. Sélectionnez le volet **Files** et dans ce volet sélectionnez le type de fichier **C++ Source File**, indiquez le nom du fichier (ex. *mon_programme_ODBC*) dans l'emplacement réservé (**File name**), puis cliquez sur OK.

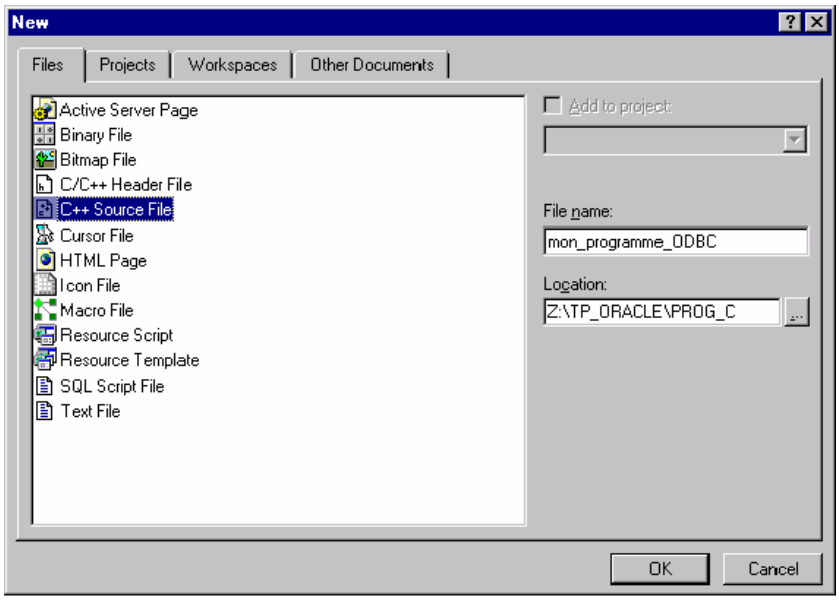

Figure 10 - Choix du type du programme sous Visual C++.

Une fenêtre s'affiche : il s'agit de l'éditeur dans lequel vous pouvez tapez le code source de votre programme.

2. Compilation du programme

Pour compiler un programme, allez dans le menu **Build** puis cliquez sur **Compile** :
S'il n'existe pas déjà, Visual vous demande si vous souhaitez créer un espace de travail (*Workspace*). Il vous suffit de répondre OUI.

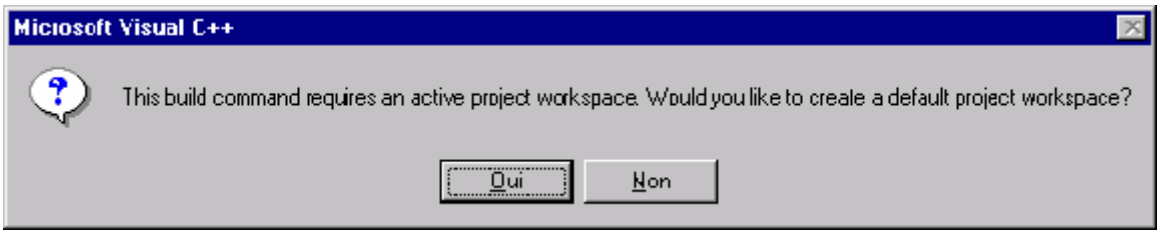

Figure 11 - Confirmation de la création d'un espace de travail.

Votre programme est compilé. Les erreurs, s'il y en a, sont affichées en bas de la fenêtre de Visual.

Si le message affiché dans la fenêtre du bas est :

« `mon_programme_ODBC.obj - 0 error(s), 0 warning(s)` »,

Vous pouvez exécuter votre programme.

Pour exécuter un programme, sélectionnez le menu **Build** puis sous-menu puis **Execute**, ou cliquez sur le point d'exclamation rouge, ou encore faites Crtl-F5. Le programme s'exécute.

C. Exemple de programme C/ODBC

1. Principe de base d'un programme ODBC

Une application ODBC possède six grandes étapes :

1. La connexion à la base de données
2. L'initialisation des paramètres d'environnement
3. L'exécution de la requête
4. Le parcours du résultat de la requête à l'aide d'un curseur
5. La validation ou l'abandon de la transaction
6. La déconnexion de la base.

Une documentation en ligne est disponible aux adresses suivantes :
http://msdn.microsoft.com/library/psdk/dasdk/odin8w4s.htm
http://msdn.microsoft.com/library/psdk/dasdk/odch4okz.htm
http://msdn.microsoft.com/library/psdk/dasdk/odap78oj.htm

Pour pouvoir utiliser les fonctions ODBC, **vous devez inclure la bibliothèques afxdb.h** dans votre programme.

Pour fonctionner, votre programme a besoin de quatre variables spécifiques :

1. Un **descripteur d'environnement**, de type HENV, qui permet d'initialiser l'environnement ODBC et d'appeler les fonctions ODBC. (pour plus de détails voir à l'adresse suivante : http://msdn.microsoft.com/library/psdk/dasdk/odch9ktv.htm)

2. Un **descripteur de connexion**, de type HDBC, qui permet de se connecter à la source de données (pour plus de détail voir à l'adresse suivante : http://msdn.microsoft.com/library/psdk/dasdk/odch3ilh.htm)

3. Un **curseur**, de type HSTMT, qui permet de se déplacer dans la table résultat de la requête.

4. D'un **code retour de fonction**, de type RETCODE, qui permet de savoir lorsqu'il y a une erreur à un moment donné de l'exécution du programme.

2. Programme exemple

Le programme, ci-après, est un programme exemple simple qui permet de se connecter à un serveur de base de données (ici, la base de données exemple), d'exécuter une requête et d'en afficher le résultat.

```
/* Inclusion des librairies */
#include <stdio.h>
#include <conio.h>
#include <afxdb.h>     // MFC ODBC database classes

char *cBASE ;          /* Nom de la source de données                    */
char *cLOGIN ;         /* Login utilisateur                              */
char *cPASSWD ;        /* Mot de passe utilisateur                      */

#define _getchar() fgetc(stdin)

void main()
{
  HENV    d_env;               /* Descripteur d'environnement            */
  HDBC    d_connex;            /* Descripteur de connexion               */
  HSTMT   curseur;             /* Curseur                                */
  RETCODE retcode;             /* Code de retour de fonction             */

  UCHAR   ucLastName[20],ucCity[20];  /* Nom et ville de l'étudiant      */
  SDWORD  ceLastName,ceCity;          /* Code d'erreur pour le nom et la */
                                      /* ville de l'étudiant             */

  char *cREQUETESQL;                  /* Variable contenant la requête   */


  /* Saisie du nom de la source de données */
  cBASE=(char*)malloc(20);
  printf("Nom de la base de donnees :");
  scanf("%s",cBASE);

  /* Saisie du login */
  cLOGIN=(char*)malloc(20);
  printf("Login :");
  scanf("%s",cLOGIN);
```

```
/* Saisie du password */
cPASSWD=(char*)malloc(20);
printf("Mot de passe : ");
/* Pour ne pas afficher le mot de passe à l'écran */
int iPosCaractere=0;
fflush(stdin);
do
{
      if((cPASSWD[iPosCaractere]=_getch())!='\r') printf("*");
}
while(cPASSWD[iPosCaractere++]!='\r' && iPosCaractere <20);
cPASSWD[--iPosCaractere]='\0';


/* Création d'un environnement ODBC :                          */
/* Allocation mémoire pour un descripteur d'environnement */
/* et initialisation l'interface d'appel d'ODBC.              */
/* Une application doit faire appel à la fonction             */
/* SQLAllocEnv pour pouvoir accéder aux fonctions ODBC.       */
retcode = SQLAllocEnv(&d_env);

/* Si la création d'un environnement ODBC est correcte */
if (retcode == SQL_SUCCESS)
{

     /* Création d'une connexion ODBC :                      */
     /* Allocation mémoire d'un descripteur de connexion  */
     /* pour l'environnement ODBC (identifié par d_env)   */
    retcode = SQLAllocConnect(d_env, &d_connex);

     /* Si la connexion ODBC s'est bien passée */
     if (retcode == SQL_SUCCESS)
     {
     /* Initialisation du temps de connexion à 5 secondes. */
     SQLSetConnectOption(d_connex, SQL_LOGIN_TIMEOUT, 5);

     /* Connexion à une source de données */
     retcode = SQLConnect(d_connex,(unsigned char*)cBASE,SQL_NTS,
                          (unsigned char*)cLOGIN,SQL_NTS,
                          (unsigned char*)cPASSWD,SQL_NTS
                         );

     /* Si la connexion à la source de données */
     /* s'est bien passée                      */
     if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
     {
        printf("Connexion a la base (source de données).\n");

        /* Pause dans l'affichage */
        printf("Taper une touche pour continuer \n"); getchar();

        /* Allocation mémoire du curseur et association du        */
        /* curseur à la source de données identifiée par d_connex.*/
        retcode = SQLAllocStmt(d_connex, &curseur);

        /* Si l'allocation mémoire du curseur est correcte */
        if (retcode == SQL_SUCCESS)
           {
             /* Creéation de la requête SQL */
             cREQUETESQL = "SELECT Nom, Ville FROM Etudiant";
```

```
            /* Execution directe de la requête sur la base */
            retcode = SQLExecDirect(curseur,
                                    (unsigned char*)cREQUETESQL,
                                     SQL_NTS
                                   );

            printf("EXECUTION DE LA REQUETE, CODE ERREUR %d,
                    CODE DE SUCCES %d \n",retcode,SQL_SUCCESS);

            /* Pause dans l'affichage */
            printf("Taper une touche pour continuer \n"); getchar();

            /* Tant le parcours du curseur est valide             */
            /* (et si l'execution de la requête s'est bien passée */
            while (retcode == SQL_SUCCESS)
             {
               /* Parcourt du résultat de la requête */
               retcode = SQLFetch(curseur);

               /* Si le parcourt  est incorrect */
               if (retcode == SQL_ERROR ||
                    retcode == SQL_SUCCESS_WITH_INFO)
                   {
                    printf("Erreur %d\n",SQL_ERROR);
                   }

            /* Si le parcourt des enregistrements est correct */
            if (retcode == SQL_SUCCESS ||
                 retcode == SQL_SUCCESS_WITH_INFO)
              {

                 /* récupération des données des colonnes 1 et 2 */
                 /* de la table résultat                         */
                 SQLGetData(curseur, 1, SQL_C_CHAR, ucLastName, 30,
                            &ceLastName);
                 SQLGetData(curseur, 2, SQL_C_CHAR, ucCity, 30,
                            &ceCity);

                 /* Affichage du résultat */
                 printf("Etudiant : %s %s\n",ucLastName,ucCity);

                 /* Pause dans l'affichage */
                 printf("Taper une touche pour continuer \n");
                 getchar();

              }


            /* Sinon, si le curseur est en fin de table */
            else
            {
                 printf("Fin\n");
            }
            } /* FIN du while */

            /* fermeture du curseur */
            SQLFreeStmt(curseur, SQL_DROP);

        } /* fin du if (retcode==SQL_SUCCESS    */
          /* || retcode==SQL_SUCCESS_WITH_INFO) */
```

```
        printf("Deconnection.\n");
        /* Deconnexion de la source de données */
        SQLDisconnect(d_connex);
        }

        /* Si la connexion ODBC n'a pu avoir lieu */
        else printf("Probleme de connexion ODBC\n");
        /* deconnexion du descripteur de connexion */
        SQLFreeConnect(d_connex);
        } /* Fin du if (retcode == SQL_SUCCESS) */

   /* désallocation mémoire de l'environnement */
   SQLFreeEnv(d_env);
   } /* Fin du premier if (retcode == SQL_SUCCESS) */
   getchar();
}
```

IV. TP SOUS POSTGRESSQL

Ce TP a pour objectif de vous faire manipuler le SGBD *PostgreSQL*.
Les scripts de la base de données exemple peuvent être retrouvés à l'adresse : http://www.lamsade.dauphine.fr/~manouvri/BD/CoursBD_MM.html

A. Création de la base exemple

1. Créer, à partir du script BDExemple.sql dont on vous donnera la localisation, le schéma de données exemple « Gestion Universitaire ».
   Attention : il faut supprimer la partie DROP du début du script, la première fois où vous créez la base.
   Une version papier du script de création de la base exemple est donnée au début de ce document (voir Section I).
2. Insérer des nuplets dans la base exemple, à partir du script BDInsertion.sql, dont on vous donnera la localisation.
3. Créer la fonction `FunctionTriggerReservation()` et le déclencheur `InsertionReservation` à partir du script FonctionEtTrigger.sql dont on vous donnera la localisation (voir Section IV.A.2).
4. Afin de vous approprier le schéma de la base et réaliser plus facilement les requêtes demandées dans la section IX.B, insérer (par la commande SQL INSERT) les nuplets suivants :
   a. Un département,
   b. Un enseignant dans le département MATHS,
   c. Un étudiant,
   d. Un enseignement de Mathématiques,
   e. Une salle,
   f. Une réservation pour un enseignement existant.

B. Interrogation de la base de données exemple

Ecrire et exécuter les requêtes SQL suivantes sur la base de données exemple :

1. Liste des noms et des prénoms des étudiants stockés dans la base.
2. Liste des noms et des prénoms des étudiants qui habitent une ville choisie dans la liste des villes de la base.
3. Liste des noms et des prénoms des étudiants dont le nom commence par 'G'
4. Liste des noms et des prénoms des enseignants dont l'avant dernière lettre du nom est 'E'.
5. Liste des noms et des prénoms des enseignants classés par nom de département, par nom et par prénom.
6. Combien y a-t-il d'enseignants dont le grade est 'Moniteur' ?
7. Quels sont les noms et les prénoms des étudiants n'ayant pas de Fax (valeur NULL)?
8. Quels sont les intitulés des enseignements dont la description contient le mot 'SQL' ou 'Licence' ?

9. Si on suppose qu'une heure d'enseignement coûte 50 euros, quel est le coût en euros de chaque enseignement (les heures de cours concernent les heures réservées – voir relation *Réservation*)?
10. A partir de la requête précédente, indiquer quels sont les intitulés des enseignements dont le coût est compris entre 500 et 750 euros.
11. Quelles sont la capacité moyenne et la capacité maximum des salles ?
12. Quelles sont les salles dont la capacité est inférieure à la capacité moyenne ?
13. Quels sont les noms et les prénoms des enseignants appartenant aux départements nommés 'MATHS' ou 'INFO' ? (utiliser *IN* puis une autre solution)
14. Quels sont les noms et les prénoms des enseignants n'appartenant ni au départements 'MATHS' ni au département 'INFO' ?
15. Regrouper les étudiants par ville.
16. Combien y a-t-il d'enseignements associés à chaque département ?
17. Quels sont les noms des départements où le nombre d'enseignements associés est supérieur ou égal à 3 ?
18. Créer une vue permettant de visualiser le nombre de réservation par enseignant.
19. Quels sont les noms et les prénoms des enseignants pour lesquels il existe au moins deux réservations ? (utiliser *EXISTS* puis une autre solution en utilisant la vue créée précédemment).
20. Quels sont les enseignants ayant le plus de réservations (Utiliser la Vue définie à la question 18 et le mot-clé ALL) ?
21. Quels sont les noms et les prénoms des enseignants n'ayant aucune réservation ?
22. Quelles salles ont été réservées à toutes les dates (stockées dans la base de données) ?
23. A quelles dates toutes les salles sont-elles réservées ?

C. Modification du schéma de la base exemple

1. Ajouter, dans la base de données exemple, une relation permettant de gérer les inscriptions des étudiants aux différents enseignements disponibles dans la base (la table doit contenir un attribut date d'inscription).
2. Ajouter, dans la base de données exemple, une relation permettant de gérer les notes des étudiants dans les différents enseignements (un étudiant peut avoir plusieurs notes pour le même enseignement).
3. Créer un déclencheur permettant de vérifier, lors de l'insertion d'une note pour un étudiant, que ce dernier possède bien une inscription pour cet enseignement (sinon ajouter l'inscription de l'étudiant à l'enseignement).

D. Accès à la base de données via ODBC

Transformer le programme C exemple donné en polycopié afin qu'il vous permette d'interroger la base de données que vous avez précédemment créée.

X. BIBLIOGRAPHIE

[GS02] *PostgreSQL - Guide du développeur* - E.Geschwinde H.Schönig - Campus Press - mai 2002 - 580 pages - ISBN :2-7440-1387-0

V. ANNEXE 1 : INSERTION DES NUPLETS DE LA BASE EXEMPLE

```
INSERT INTO Departement VALUES ('1','INFO');
INSERT INTO Departement VALUES ('2','MATHS');
INSERT INTO Departement VALUES ('3','GESTION');
INSERT INTO Departement VALUES ('4','ECO');
INSERT INTO Departement VALUES ('5','LANGUES');

INSERT INTO Etudiant VALUES ('1','GAMOTTE', 'Albert','18/02/1979','50, Rue des
alouettes','PARIS','75021','0143567890',NULL,'gamotal4@etud.dauphine.fr');
INSERT INTO Etudiant VALUES ('2','HIBULAIRE', 'Pat','23/08/1980','10, Avenue des
marguerites','POUILLON','40000','0678567801',NULL,'pat@yahoo.fr');
INSERT INTO Etudiant VALUES ('3','ODENT', 'Jamal','05/12/1978','25, Boulevard des
fleurs','PARIS','75022','0145678956','0145678956','odent@free.fr');
INSERT INTO Etudiant VALUES ('4','RASLATABLE', 'Deborah','15/07/1979','56, Boulevard des
fleurs','PARIS','75022','0678905645',NULL,'deby@hotmail.com');
INSERT INTO Etudiant VALUES ('5','Debécé', 'Aude','15/08/1979','45, Avenue des
abeilles','PARIS','75022',NULL,NULL,NULL);

INSERT INTO Enseignant
VALUES('1','1','MANOUVRIER','Maude','MCF','4185','4091','manouvrier@lmasade.dauphine.fr');
INSERT INTO Enseignant
VALUES('2','1','NAIJA','Yosr','Moniteur','4917','4091','naija@lmasade.dauphine.fr');
INSERT INTO Enseignant
VALUES('3','1','BAHRI','Afef','Moniteur','4917','4091','bahri@lmasade.dauphine.fr');
INSERT INTO Enseignant VALUES('4','1','LIMAM','Medhi','ATER',NULL,NULL,NULL);
INSERT INTO Enseignant VALUES('5','5','MyTaylor','IsRich','Vacataire',NULL,NULL,NULL);

INSERT INTO Salle VALUES('B','020','15');
INSERT INTO Salle VALUES('B','022','15');
INSERT INTO Salle VALUES('A','301','45');
INSERT INTO Salle VALUES('C','Amphi 8','500');
INSERT INTO Salle VALUES('C','Amphi 4','200');


INSERT INTO Enseignement VALUES ('1','1','Bases de Données','Niveau Licence : Modélisation E/A
et UML, Modèle relationnel, Algèbre Relationnelle, Calcul relationel, SQL, dépendances
fonctionnelles et formes normales');
INSERT INTO Enseignement VALUES ('2','1','Mise à Niveau Informatique','Pour les étudiants de
GMI entrant directement en IUP2: Architecture, Algorithmique, Langage C et Graphes');
INSERT INTO Enseignement VALUES ('3','1','Mise à Niveau Bases de Données','Pour les étudiants
de DESS ID ou DEA127 - Programme Licence et Maîtrise en Bases de Données');
INSERT INTO Enseignement VALUES ('4','5','Anglais','');

INSERT INTO Reservation VALUES
('1','B','022','1','1','1','15/10/2003','08:30:00','11:45:00','3');
INSERT INTO Reservation VALUES
('2','B','022','1','1','2','04/11/2003','08:30:00','11:45:00','3');
INSERT INTO Reservation VALUES
('3','B','022','1','1','2','07/11/2003','08:30:00','11:45:00','3');
INSERT INTO Reservation VALUES
('4','B','020','1','1','2','20/10/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('5','B','020','1','1','3','09/12/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('6','A','301','2','1','1','02/09/2003','08:30:00','11:45:00','3');
INSERT INTO Reservation VALUES
('7','A','301','2','1','1','03/09/2003','08:30:00','11:45:00','3');
INSERT INTO Reservation VALUES
('8','A','301','2','1','1','10/09/2003','08:30:00','11:45:00','3');
INSERT INTO Reservation VALUES
('9','A','301','3','1','1','24/09/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('10','B','022','3','1','1','15/10/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('11','A','301','3','1','1','01/10/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('12','A','301','3','1','1','08/10/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('13','B','022','1','1','4','03/11/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('14','B','022','1','1','3','20/10/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('15','B','022','1','1','2','09/12/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('16','B','022','1','1','4','03/09/2003','08:30:00','11:45:00','3');
INSERT INTO Reservation VALUES
('17','B','022','1','1','2','10/09/2003','08:30:00','11:45:00','3');
```

```
INSERT INTO Reservation VALUES
('18','B','022','1','1','4','24/09/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('19','B','022','1','1','3','01/10/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('20','B','022','1','1','1','08/10/2003','13:45:00','17:00:00','3');
INSERT INTO Reservation VALUES
('21','B','022','1','1','4','02/09/2003','08:30:00','11:45:00','3');
```

VI. ANNEXE 2 : INSTALLATION DE POSTGRESQL SOUS WINDOWS

1. Installer l'outil *cygwin* (émulateur linux/unix) à partir de l'adresse http://www.cygwin.com/setup.exe en indiquant bien d'installer les packages *cygpic* (*IPC support for cygwin*) et *postgressql* (*PostgreSQL Data Base Management System*)

2. Une fois cygwin installé, dans cygwin (cf. document INSTALL dans le répertoire `cygwin/usr/doc/postgressql-7.3.4` :

   - Taper la commande ipc-daemon2 & (pour appeler les librairies utilisées par postgres)
   - S'il n'existent pas, créer les répertoires `/usr/localpgsql` et `/usr/local/pgsql/data`
     `mkdir /usr/local/pgsql` et `mkdir /usr/local/pgsql/data`
   - Taper la commande `/usr/bin/initdb -D /usr/local/pgsql/data`
   - Puis, taper la commande
     `/usr/local/pgsql/bin/postmaster –i -D /usr/local/pgsql/data`
     (sans oublier le –i)

3. Installer l'interface graphique *PgAdmin* à partir de l'adresse http://pgadmin.postgresql.org/pgadmin3/download.php ou ftp://ftp2.fr.postgresql.org/ et aller dans le répertoire `pgadmin3/beta/win32` pour récupérer le fichier .zip

Attention :

A chaque fois que vous voudrez utiliser *postgres* il faudra refaire sous *cygwin* les commandes :

```
ipc-daemon2 &

/usr/local/pgsql/bin/postmaster –i -D /usr/local/pgsql/data
```