Licence de Mathématiques, Informatique et Applications à l'Économie et à l'Entreprise (MI2E)
Mention Mathématiques Mineur Informatique et Mention Informatique
3ème année

ANNEE 2012 / 2013

Désignation de l'enseignement : Bases de données relationnelles

Nom du document : TP PostgreSQL

Rédacteur : Maude Manouvrier

TABLE DES MATIERES

PostgreSQL est un Système de Gestion de Bases de Données Relationnel Objet, *open source*, successeur 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`
FAQ en Français : `http://www.postgresql.org/docs/faqs.FAQ_french.html`
Documentation en français : `http://docs.postgresqlfr.org/`
Site de la communauté française : `http://www.postgresql.fr/`

Ce document a pour objectif de vous aider à utiliser ce SGBD. Il contient le sujet du TP (voir Section I), le script de la base de données exemple (voir Section II), le scripts d'insertion des nuplets (voir Section VIII), des exemples de fonctions SQL, PL/pgSQL et de déclencheur (voir Section III), une description de l'interface de l'outil PgAdmin (voir Section V), un exemple de programme ODBC (voir Section VI) et quelques informations sur JDBC (voir Section VI). Il contient également un petit guide d'installation de PosgreSQL sous Windows (voir Section VIII).

I. ENONCE DU TP SOUS POSTGRESSQL

Ce TP a pour objectif de vous faire manipuler le SGBD *PostgreSQL* et le langage SQL.

Les scripts SQL de la base de données exemple peuvent être retrouvés à l'adresse : `http://www.lamsade.dauphine.fr/~manouvri/TPBD/`

A. Première utilisation de PostgreSQL et connexion à la base de données

1. Lancer  : Menu **Applications**, puis **Outils Système**, puis **PgAdmin III**- Voir Section V page 12 pour plus de détails sur l'interface.
2. Connecter vous à la base en cliquant sur le bouton représentant une prise de courant en haut à gauche. Une fenêtre apparaît.
3. Taper `database.etud.dauphine.fr` dans le champ **Hôte** de la fenêtre: il s'agit de l'adresse du serveur sur lequel à été installé le SGBD PostgreSQL.
4. Taper `Crio UNIX` dans le champ **Nom** : le bouton OK n'est plus grisé.
5. Taper votre *login* dans le champ **BD maintenance** : la base de données, sur laquelle vous allez travailler, a le même nom que votre nom d'utilisateur.
6. Taper votre *login* dans le champ **Nom Utilisateur** et votre **Mot de Passe** dans le champ correspondant et cliquer sur OK.
7. Cliquer sur le symbole + situé à coté du terme **Bases de Données** dans le menu à gauche, puis sur le symbole + situé à coté de la base de données correspondant à votre *nom d'utilisateur*.

B. Lancer l'interpréteur SQL

Cliquer sur le bouton contenant le mot SQL situé dans la barre d'icônes en haut de la fenêtre de PgAdmin3 pour lancer l'interpréteur de requêtes `SQL` – Voir Section V page 12 pour plus de détails sur l'interface. Il s'agit du principal outil que vous utiliserez pendant ce TP, l'objectif de ce TP étant le langage SQL.

C. Création de la base exemple

1. Créer le schéma de données de la base exemple en exécutant le script `BDExemple.sql` dont on vous donnera la localisation[1] dans l'interpréteur de requêtes SQL.

   Il vous suffit de recopier le contenu du fichier dans la fenêtre du haut de l'interpréteur. Vous pouvez aussi sauvegarder le fichier dans votre répertoire et l'ouvrir dans l'interpréteur SQL.

   Attention : L'interface de la version de PgAdmin3, installée au CRIO UNIX, a un bug. Si vous ouvrez un fichier dans l'interpréteur SQL, rien ne se passe. IL faut ouvrir les fichiers dans un éditeur de texte quelconque et copier le contenu dans l'interpréteur SQL.

   Une version papier du script de création de la base exemple est donnée dans Section II (voir page 6). Une partie de ce script vous a été décrit en cours.

[1] `http://www.lamsade.dauphine.fr/~manouvri/TPBD/BDExemple.sql`

Pour exécuter le script SQL il suffit de cliquer sur le bouton représentant un triangle vert  en haut de l'interpréteur SQL.

2. Insérer les nuplets exemples, en exécutant le script d'insertion `BDInsertion.sql`, dont on vous donnera la localisation[2]. Une version papier du script vous est donné dans la section VIII page 21.
3. Exécuter le script `FonctionEtTrigger.sql` dont on vous donnera la localisation[3] (voir version papier à la Section III page 9) pour créer la fonction `FunctionTriggerReservation()` ainsi que le déclencheur `InsertionReservation`.
4. Afin de vous approprier le schéma de la base et réaliser plus facilement les requêtes demandées dans la section suivante, insérer (par la commande `SQL INSERT`) les nuplets suivants :
    a. Un département,
    b. Un enseignant dans le département MIDO,
    c. Un étudiant,
    d. Une salle,
    e. Une réservation pour un enseignement existant,
    f. Une réservation qui chevauche une réservation existante (pour tester l'affichage du déclencheur).

NB : Si vous sélectionnez plusieurs lignes avant de cliquer sur le bouton  , seules les lignes sélectionnées seront exécutées.

Pour mettre des lignes en commentaires, vous devez précéder chaque ligne par – (deux tirets) ou placer les lignes à mettre en commentaire entre /* et */.

Vous pouvez reprendre et suivre le polycopié initial.

D. Interrogation de la base de données exemple

Ecrire et exécuter les requêtes[4] 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 (par vous) 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)?

[2] `http://www.lamsade.dauphine.fr/~manouvri/TPBD/BDInsertion.sql`
[3] `http://www.lamsade.dauphine.fr/~manouvri/TPBD/FonctionEtTrigger.sql`

[4] Attention : toutes les requêtes ne retournent pas forcement de résultat. Certaines peuvent retourner une relation vide (i.e. sans nuplet). Vous pouvez insérer des nuplets en conséquence pour qu'il y ait des nuplets résultat.

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 'MIDO' ou 'LSO' ? (utiliser *IN* puis une autre solution)
14. Quels sont les noms et les prénoms des enseignants n'appartenant ni au département 'MIDO' ni au département 'LSO' ?
15. Classer 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 ?

E. 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).

II. 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. *Ces commandes sont mises en commentaires (précédées de deux tirets --), car elles sont inutiles à la première création de la base.*

```
-- Suppression des relations si elles sont deja créées
-- (enlever les tirets de commentaires)
-- L'ordre de suppression des relations doit être respecté
-- pour ne pas violer les contraintes d'intégrité référentielles
-- DROP VIEW Email_Etudiant;
-- DROP TABLE Etudiant;
-- DROP TABLE Reservation;
-- DROP FUNCTION GetSalleCapaciteSuperieurA(int);
-- 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;
```

Remarque : vous pouvez choisir d'utiliser une incrémentation automatique des clés primaires, lorsqu'elles sont mono-attribut et de type entier. Il faut pour cela utiliser le type **SERIAL**. Par exemple, l'instruction SQL suivante crée une relation `TableEssai` dont l'attribut `idAuto` est un entier qui s'incrémente à chaque insertion de nuplet.

```
CREATE TABLE TableEssai (idAuto SERIAL, nom VARCHAR(100));
```

Cette instruction implique la création implicite d'une séquence, comme l'indique le message affiché par le SGBD suite à la commande de création de la relation.

```
=>NOTICE:CREATE TABLE will create implicit sequence
"tableessai_idauto_seq" for serial column "tableessai.idauto"
```

Le type de l'attribut `idAuto` sera le type entier et cet attribut aura une valeur égale par défaut à `nextval('tableessai_idauto_seq')`.

Une insertion dans cette relation devra par conséquent se faire de la manière suivante :

```
INSERT INTO TableEssai
 VALUES (nextval('tableessai_idauto_seq'),'Toto');
```

III. 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 `GetSalleCapaciteSuperieurA` 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 `GetDepartement_ID` 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 'MIDO'.

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

```
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 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 `PossibiliteResa` vérifie que le créneau horaire choisi 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. Elle 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 2006 entre 9h et 18h.

```
SELECT PossibiliteResa('B','022','2006/11/04','09:00:00','18:00:00');
```

B. Fonction trigger en PL/pgSQL

1. Langage 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.

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
```

← *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)*

```
        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
           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;   ← Si on peut faire l'insertion, la fonction retourne le nuplet
  END IF;              en cours d'insertion, représenté par NEW
END;'
LANGUAGE 'plpgsql';
```

La fonction `FunctionTriggerReservation` 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 `InsertionReservation` s'exécute avant l'insertion de tout nuplet dans la table *Réservation*. Il fait appel à la fonction `InsertionReservation`, expliquée précédemment.

V. ANNEXE 1 : INTERFACE DE L'OUTIL PGADMIN

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

`PgAdmin III` est un outil graphique permettant de manipuler PostgreSQL. Pour plus d'informations, vous pouvez consulter l'adresse : `http://www.pgadmin.org/` et `http://www.pgadmin.org/?locale=fr_FR`

A. Lancer PgAdmin

Pour accéder au logiciel depuis le CRIO UNIX, aller menu **Applications**, le sous-menu **Programmation**, sous-menu **pgAdmin III**.
La fenêtre équivalente à la fenêtre suivante s'affiche :

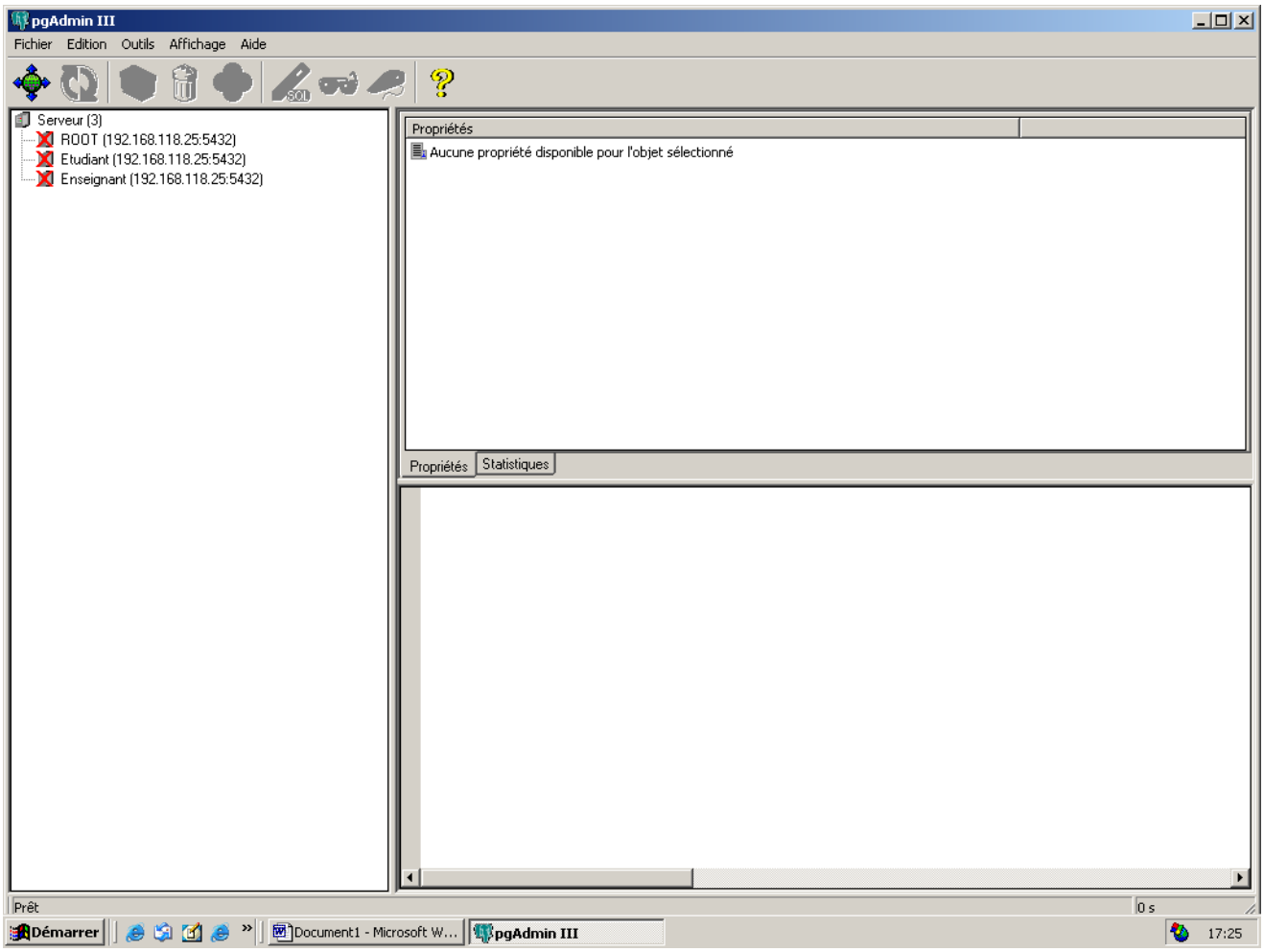

Figure 1 - Fenêtre principale de PgAdmin III.

B. Se connecter

Pour vous connecter, cliquez sur le bouton vert en haut à gauche (sous le menu principal), puis cliquez sur la base de données portant votre nom d'utilisateur et choisissez **Connexion** (ou double-cliquez sur votre nom utilisateur)**.** Une fenêtre apparaît. Dans cette fenêtre, tapez le numéro IP du serveur (ex. `database.etud.dauphine.fr`), votre login et votre mot de passe et cliquez sur OK.

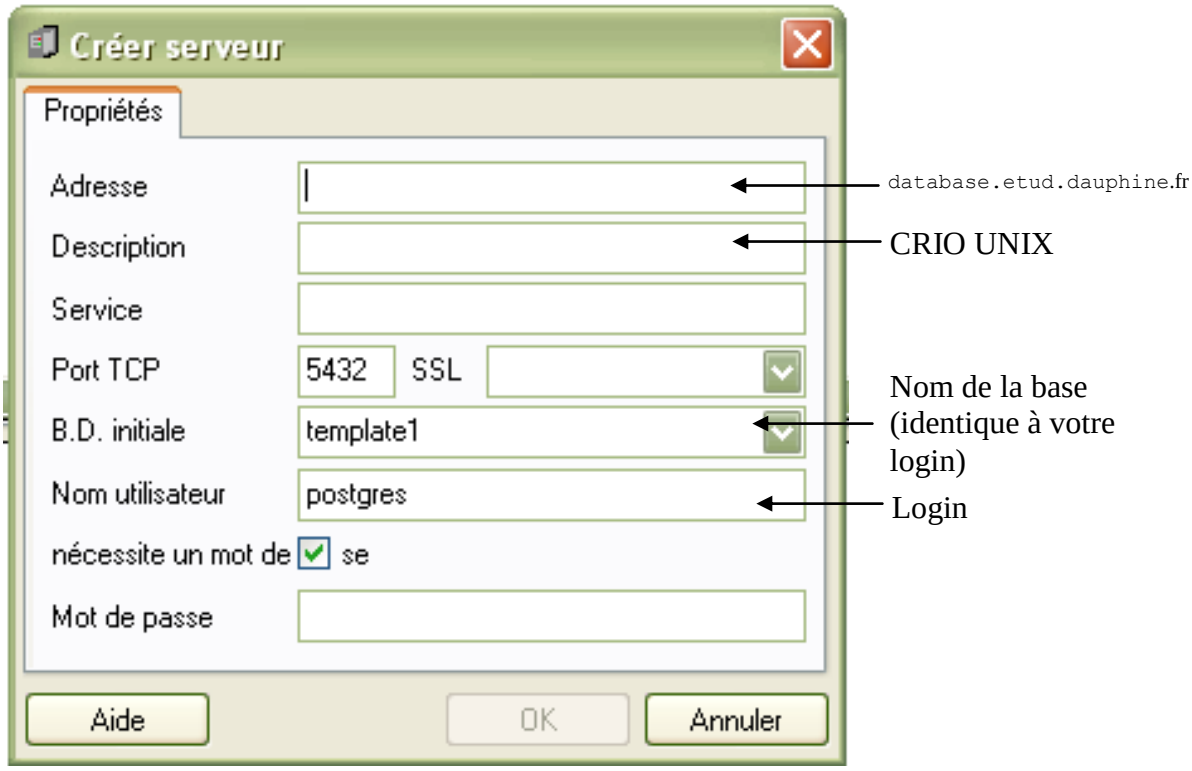

Figure 2 – Connexion à un serveur pour une base de données.

C. Exécuter des commandes SQL

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

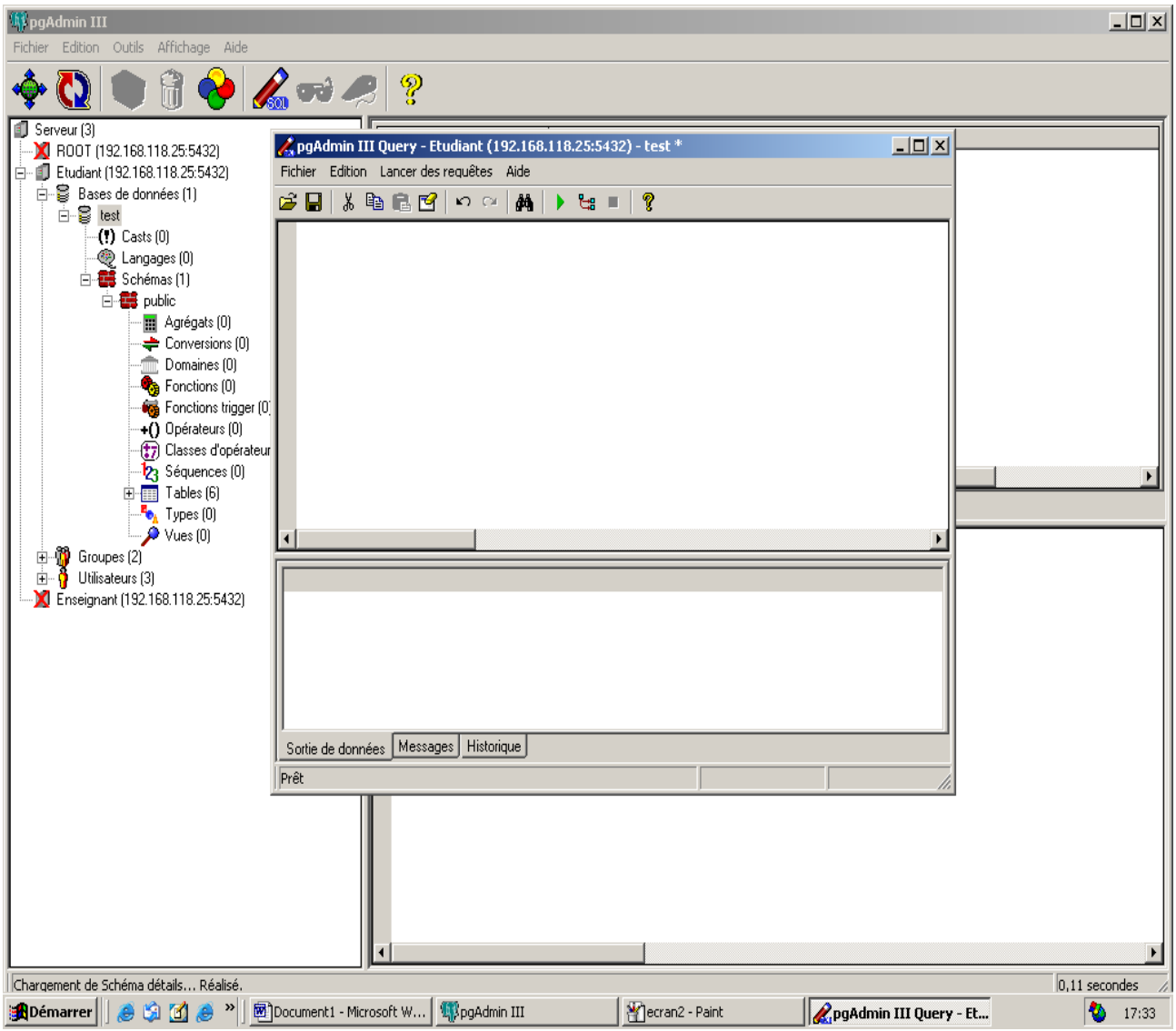

Figure 3 - Interpréteur de commandes SQL de PgAdmin III.

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** (ou cliquez sue la flèche verte).

Les commentaires dans une requête commencent par – – (deux tirets).

Vous pouvez charger le contenu d'un fichier dans la fenêtre à partir du menu.

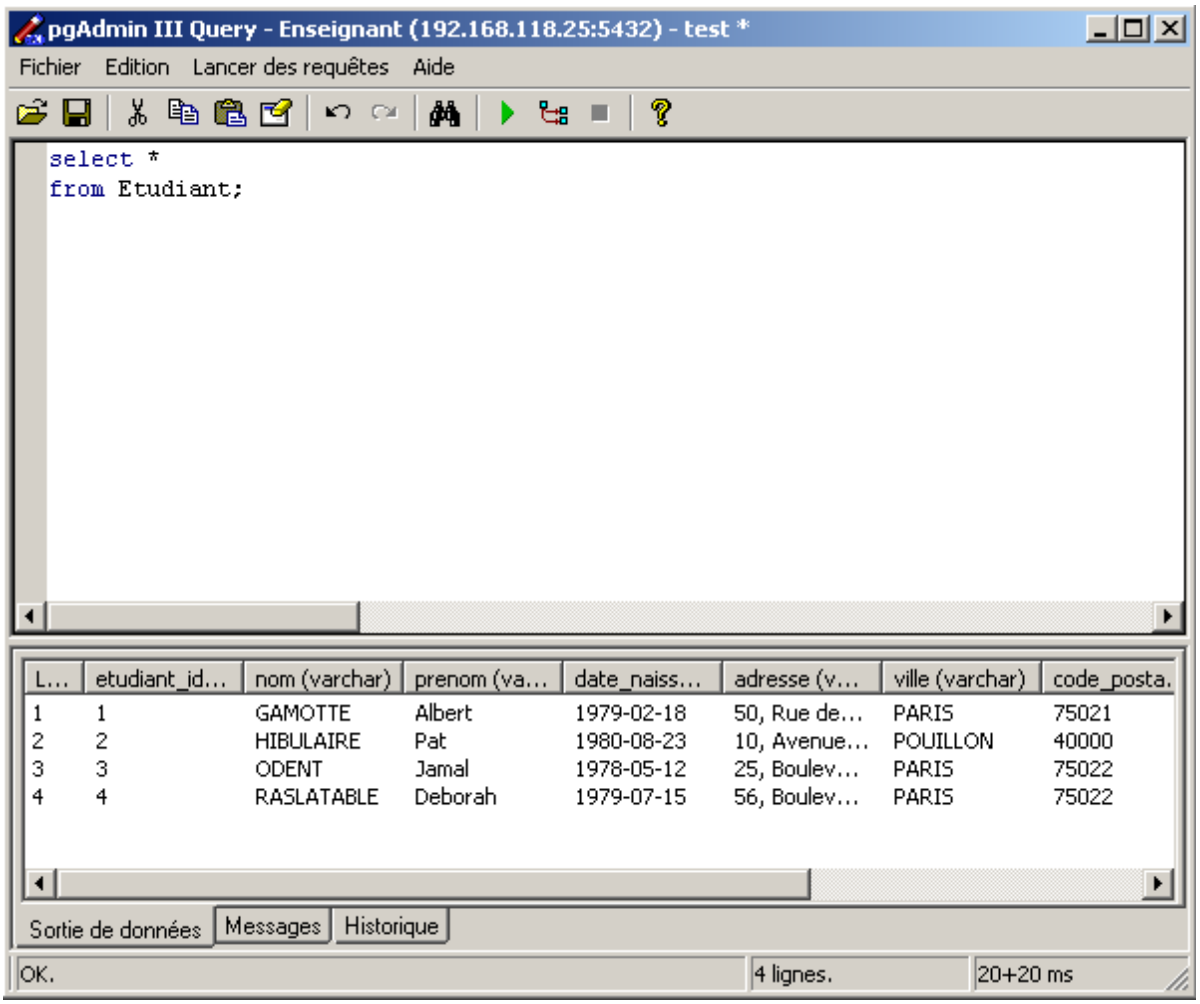

Figure 4 - Exemple de requête exécutée dans l'interpréteur de commandes SQL.

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 III. Lorsque vous cliquez sur le nom d'une table, son schéma au format SQL apparaît en bas à droite de la fenêtre.

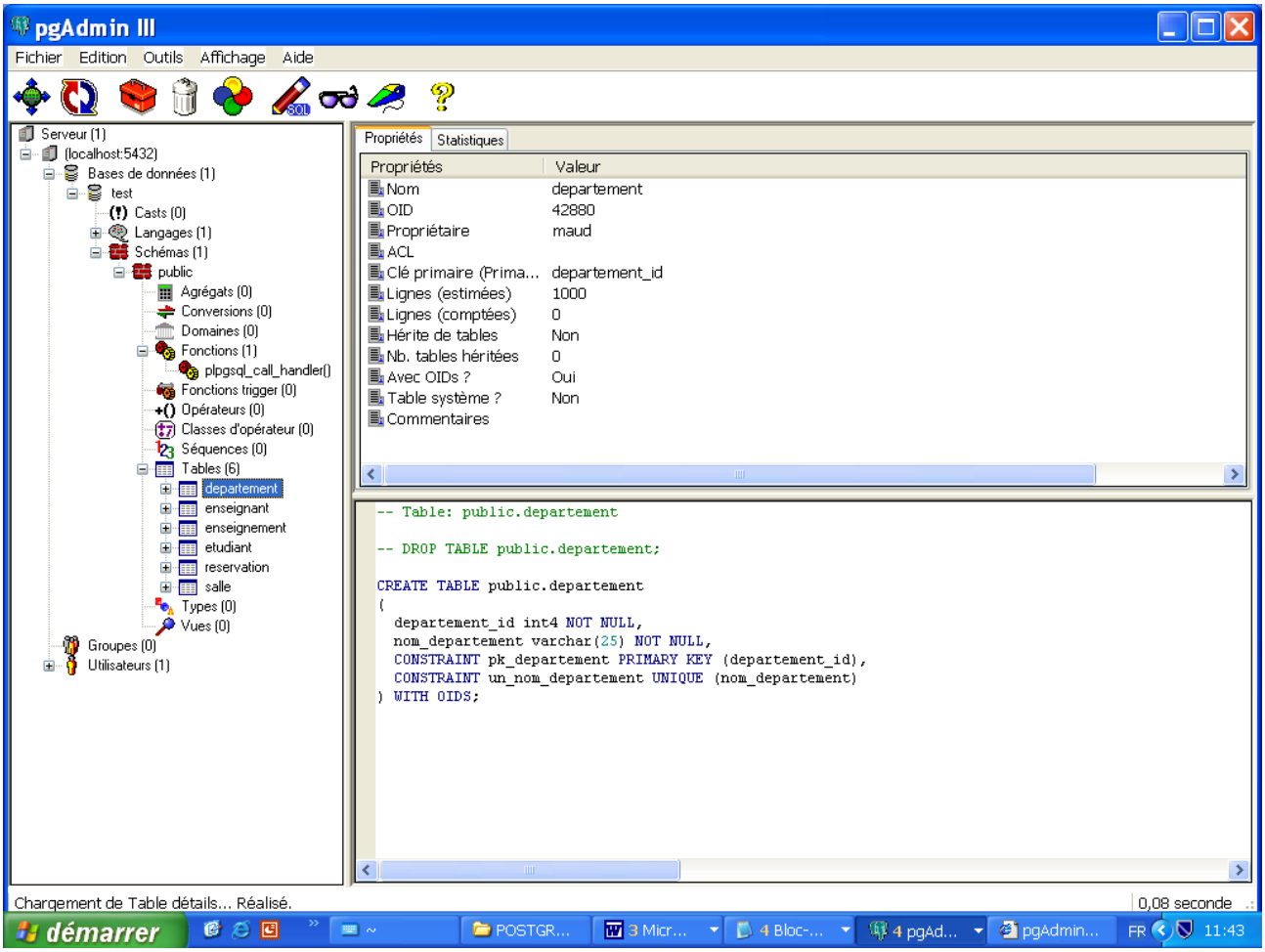

Figure 5 - Visualisation du script SQL de création de la relation *Département.*

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 III (en particulier à l'aide des commandes SQL) via le menu ou en appuyant sur la touche *F1*.

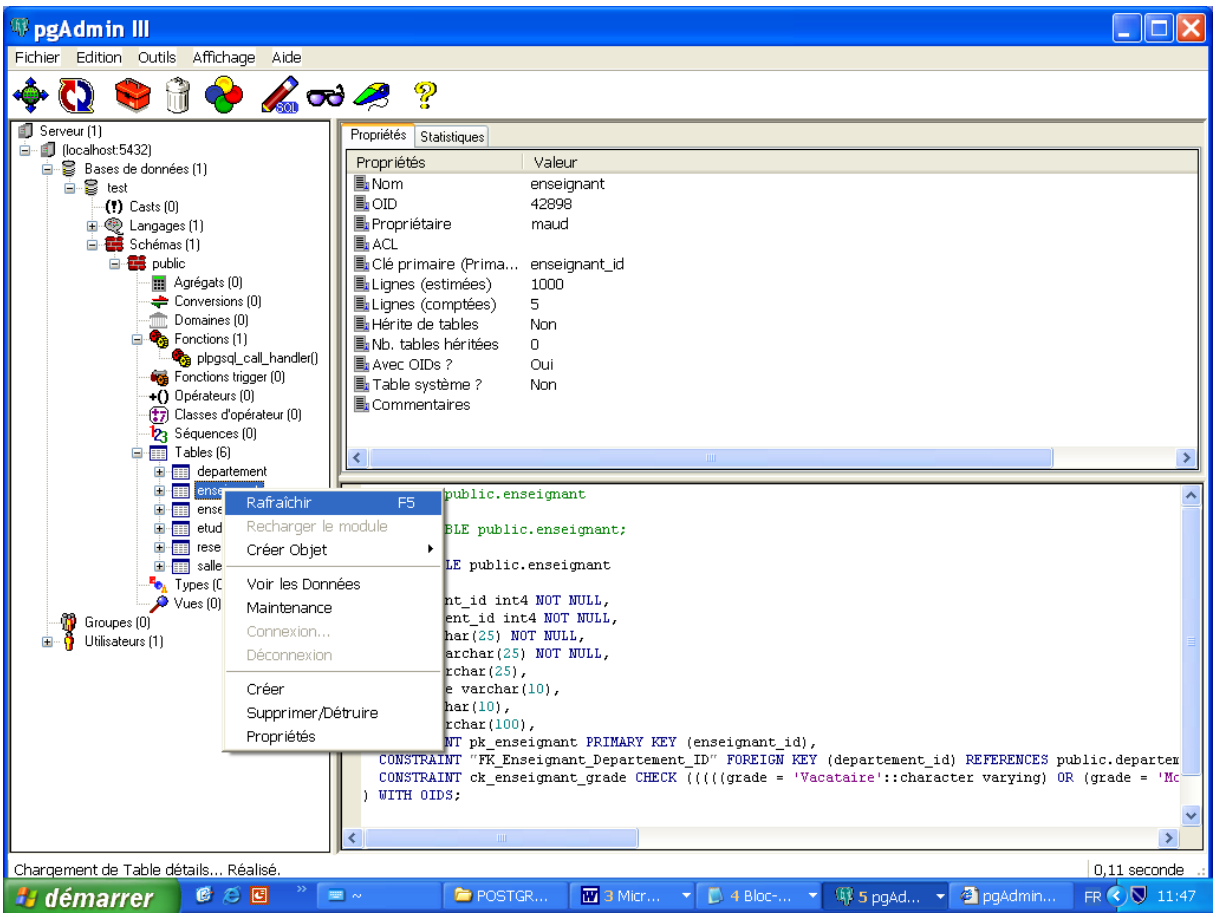

Figure 6 - Accès aux propriétés d'une relation (en particulier accès aux données).

VI. ANNEXE 2 : EXEMPLES DE PROGRAMMES JDBC/ODBC

JDBC/ODBC est un *middelware* facilitant la connexion entre un client de bases de données (fonctionnant uniquement sous Windows[5] pour ODBC) et un serveur de base de données. 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 JDBC/ODBC**, peut interroger différentes bases de données sur différentes plates-formes. Le langage de l'API d'JDBC/ODBC est une combinaison d'appels systèmes et de SQL. Il existe pour chaque SGBD, un **pilote JDBC/ODBC** (ou *driver*) particulier. Ce pilote permet la traduction des commandes JDBC/ODBC en commandes spécifiques au SGBD utilisé.

A. Exemple de programme JDBC

Vous pouvez trouver le driver jdbc pour PostgreSQL à l'adresse suivante : `http://jdbc.postgresql.org/download.html`

Une documentation est disponible à l'adresse : http://jdbc.postgresql.org/doc.html. Une liste de diffusion est disponible à l'adresse : `http://archives.postgresql.org/pgsql-jdbc/`

Le code source des programmes exemples sont disponibles à l'adresse : http://www.lamsade.dauphine.fr/~manouvri/HIBERNATE/TP_JDBC/TP_JDBC.html

Le fichier `TestJDBCPostgresql.java` est un exemple complet (il montre comment se connecter, comment exécuter des requêtes de mise à jour et de sélection). Le fichier `FichierConnexion.txt` contient les paramètres de connexion (nom du pilote et adresse de la base). Ce fichier permet de ne pas modifier le programme à chaque changement de base ou de SGBD (voir commentaire du fichier `TestJDBCPostgresql.java`).

Le fichier `Departement.java` est un exemple de classe Java dont les objets sont persistants (i.e. sont récupérés à partir de données de la base de données ou dont les valeurs des attributs sont stockées dans la base). Le fichier `CreerDepartement.java` permet de tester cette classe.

Ces programmes ont été adaptés à la base de données exemple à partir des exemples de http://www.fankhausers.com/postgresql/jdbc/#driver_download et de http://deptinfo.unice.fr/~grin/mescours/minfo/bdavancees/tp/tpjdbc1/index.html

Si vous utilisez le SGBD PostgreSQL, le pilote JDBC se nomme `org.postgresql.Driver` et l'adresse de la base de données au CRIO UNIX est la suivante : jdbc:postgresql://database.etud.dauphine.fr/nom_base

B. Connexion d'une source de données à ODBC

Sous ODBC, la gestion est un peu différente**,** on utilise **un gestionnaire de source de données. 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

[5] 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 :

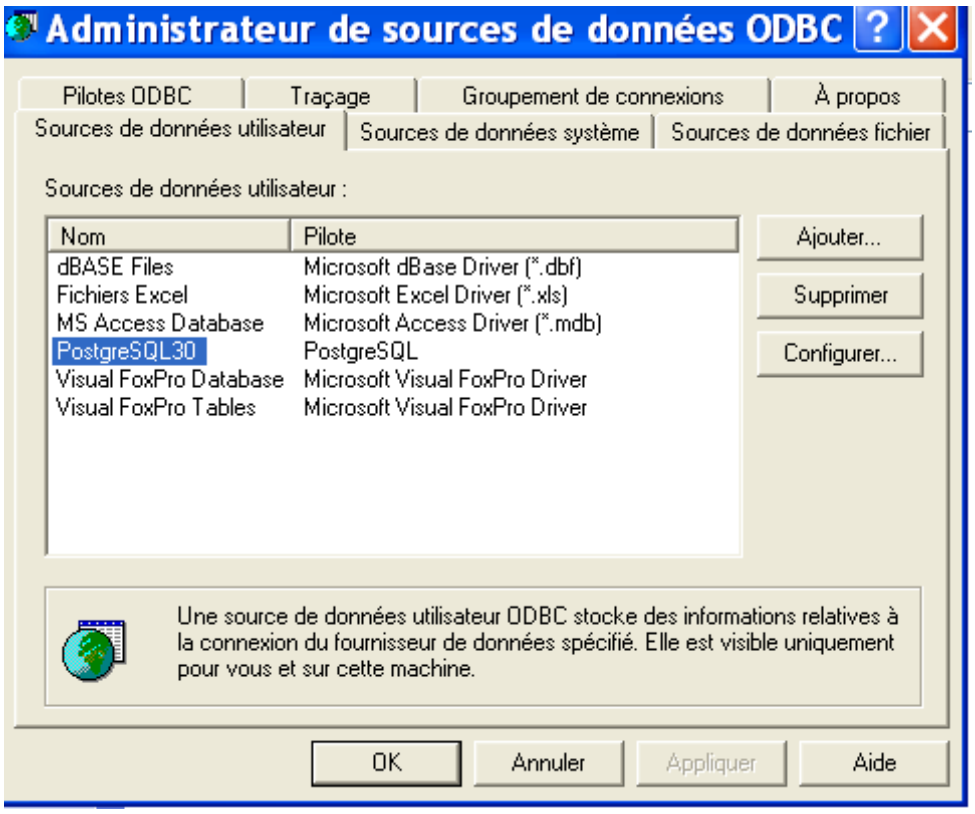

Figure 8 - Choix d'une source de données ODBC.

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 :

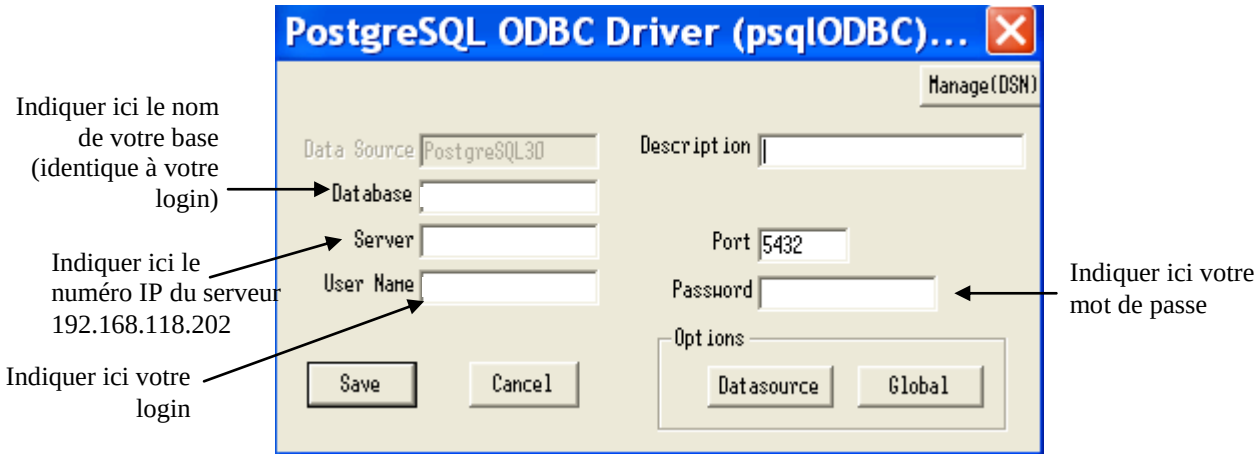

Figure 9 - Fenêtre de source de données ODBC propre à PosgreSQL.

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

Une documentation en ligne est disponible aux adresses suivantes :
`msdn.microsoft.com/library/FRE/vccore/html/_core_database_topics_.28.odbc.29.asp`
`http://msdn.microsoft.com/library/fre/default.asp?url=/library/FRE/vccore/html/_core_odbc.asp`

Pour pouvoir utiliser les fonctions ODBC, vous devez inclure la bibliothèque de la PI ODBC (ex. `afxdb.h` sous Visual C++) 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.

2. Un descripteur de connexion, de type HDBC, qui permet de se connecter à la source de données

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.

Le code source du programme exemple (adapté du tutoriel Visual C++ à la base de données exemple) est disponible à l'adresse :
http://www.lamsade.dauphine.fr/~manouvri/BD/prog_ex_ODBC.cpp

VIII. Annexe 4 : insertion des nuplets de la base exemple

```
INSERT INTO Departement VALUES ('1','MIDO');
INSERT INTO Departement VALUES ('2','LSO');
INSERT INTO Departement VALUES ('3','MSO');
INSERT INTO Departement VALUES ('4','LANGUES');

INSERT INTO Etudiant VALUES ('1','GAMOTTE', 'Albert','1979/02/18','50, Rue des
alouettes','PARIS','75021','0143567890',NULL,'gamotal4@etud.dauphine.fr');

INSERT INTO Etudiant VALUES ('2','HIBULAIRE', 'Pat','1980/08/23','10, Avenue des
marguerites','POUILLON','40000','0678567801',NULL,'pat@yahoo.fr');

INSERT INTO Etudiant VALUES ('3','ODENT', 'Jamal','1978/05/12','25, Boulevard des
fleurs','PARIS','75022','0145678956','0145678956','odent@free.fr');

INSERT INTO Etudiant VALUES ('4','SLATABLE', 'Deborah','1979/07/15','56, Boulevard des
fleurs','PARIS','75022','0678905645',NULL,'deby@hotmail.com');

INSERT INTO Etudiant VALUES ('5','DEBECE', 'Aude','1979/08/15','45, Avenue des
abeilles','PARIS','75022',NULL,NULL,NULL);

INSERT INTO Enseignant
VALUES('1','1','MANOUVRIER','Maude','MCF','4185','4091','manouvrier@lamsade.dauphine.fr');

INSERT INTO Enseignant VALUES('2','1','PABIEN','Yvon','Moniteur', NULL, NULL, NULL);

INSERT INTO Enseignant VALUES('3','1','DEBECE','Gil','Moniteur', NULL, NULL, NULL);

INSERT INTO Enseignant VALUES('4','1','DEPEN','Amédé','PROF', NULL, NULL, NULL);

INSERT INTO Enseignant
VALUES('5','1','SUFFIT','Sam','Moniteur',NULL,NULL,'guehis@lamsade.dauphine.fr');

INSERT INTO Enseignant VALUES('6','4','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 Relationnelles','Niveau Licence
(L3) : Modélisation E/A et UML, Modèle relationnel, Algèbre Relationnelle, Calcul relationel,
SQL, dépendances fonctionnelles et formes normales');

INSERT INTO Enseignement VALUES ('2','1','Langage C++','Niveau Master 1');

INSERT INTO Enseignement VALUES ('3','1','Mise à Niveau Bases de Données','Niveau Master 2 -
Programme Licence et Master 1 en Bases de Données');

INSERT INTO Enseignement VALUES ('4','4','Anglais','');

INSERT INTO Reservation VALUES
('1','B','022','1','1','1','2006/10/15','08:30:00','11:45:00','3');

INSERT INTO Reservation VALUES
('2','B','022','1','1','4','2006/11/04','08:30:00','11:45:00','3');

INSERT INTO Reservation VALUES
('3','B','022','1','1','4','2006/11/07','08:30:00','11:45:00','3');

INSERT INTO Reservation VALUES
('4','B','020','1','1','5','2006/10/20','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('5','B','020','1','1','4','2006/12/09','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('6','A','301','2','1','1','2006/09/02','08:30:00','11:45:00','3');

INSERT INTO Reservation VALUES
('7','A','301','2','1','1','2006/09/03','08:30:00','11:45:00','3');

INSERT INTO Reservation VALUES
('8','A','301','2','1','1','2006/09/10','08:30:00','11:45:00','3');

INSERT INTO Reservation VALUES
('9','A','301','3','1','1','2006/09/24','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('10','B','022','3','1','1','2006/10/15','13:45:00','17:00:00','3');
```

```
INSERT INTO Reservation VALUES
('11','A','301','3','1','1','2006/10/01','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('12','A','301','3','1','1','2006/10/08','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('13','B','022','1','1','4','2006/11/03','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('14','B','022','1','1','5','2006/10/20','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('15','B','022','1','1','4','2006/12/09','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('16','B','022','1','1','4','2006/09/03','08:30:00','11:45:00','3');

INSERT INTO Reservation VALUES
('17','B','022','1','1','5','2006/09/10','08:30:00','11:45:00','3');

INSERT INTO Reservation VALUES
('18','B','022','1','1','4','2006/09/24','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('19','B','022','1','1','5','2006/10/01','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('20','B','022','1','1','1','2006/10/08','13:45:00','17:00:00','3');

INSERT INTO Reservation VALUES
('21','B','022','1','1','4','2003/09/02','08:30:00','11:45:00','3');
```

VIII. ANNEXE 5 : INSTALLATION DE POSTGRESQL SOUS WINDOWS

Pour installer POSTGRESQL sous WINDOWS, des informations sont disponibles sur la page Web : http://www.lamsade.dauphine.fr/~manouvri/BD/CoursBD_MM.html

D'autres documentations sont également disponibles sur :
http://docs.postgresql.fr/8.3/INSTALL.html
http://pginstaller.projects.postgresql.org/
http://pgfoundry.org/projects/pginstaller
http://www.postgresql.org/