

Licence Mathématiques et Informatique 3ème année

ANNEE 2023/ 2024

Désignation de l'enseignement : SQL

Nom du document : TP SQL sous PostgreSQL

Rédacteur : Maude Manouvrier

La reproduction de ce document par tout moyen que ce soit est interdite conformément
aux articles L111-1 et L122-4 du code de la propriété intellectuelle

TABLE DES MATIERES

I.	DESCRIPTION DE POSTGRESQL	4
II.	CREATION DE VOTRE BASE DE DONNEES EN UTILISANT LE SGBD POSTGRESQL INSTALLE A DAUPHINE	4
III.	UTILISATION DE POSTGRESQL A DAUPHINE.....	4
V.	SCRIPT DE LA BASE DE DONNEES EXEMPLE	7
VI.	INSERTION DES NUPLETS DE LA BASE EXEMPLE	10
VII.	EXEMPLE DE FONCTIONS SQL, PL/PGSQL ET DE DECLENCHEUR SOUS POSGRESQL	11
A.	FONCTIONS SQL	11
B.	FONCTION TRIGGER EN PL/PGSQL	12
1.	<i>Langage PL/pgSQL.....</i>	<i>12</i>
2.	<i>Exemple de fonction utilisée dans un déclencheur.....</i>	<i>13</i>
C.	TRIGGER	13
VIII.	TRAVAIL A FAIRE PENDANT LE TP.....	14
A.	CREATION DE LA BASE EXEMPLE.....	14
B.	INTERROGATION DE LA BASE DE DONNEES EXEMPLE	14
C.	MODIFICATION DU SCHEMA DE LA BASE EXEMPLE.....	15
IV.	DESCRIPTION DE DBFIDDLE	15
V.	DESCRIPTION D'UN SITE <i>BAC A SABLE</i> PERMETTANT D'UTILISER POSTGRESQL.....	16
IX.	ANNEXE : INSTALLATION DE POSTGRESQL SUR VOTRE MACHINE	17
X.	ANNEXE : EXEMPLES DE PROGRAMMES JDBC	18

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

Ce document contient une description des outils utilisés (voir Sections I à IV), la description du script de la base de données exemple le sujet du TP (voir Section V), des exemples de fonctions SQL, PL/pgSQL et de déclencheur le script d'insertion des nuplets (voir Section VI et Section VII). En annexe, vous trouverez des informations sur l'installation de PostgreSQL et quelques informations sur JDBC (pour accéder à une base de données dans un programme Java).

I. Description de POSTGRESQL

Le SGBD utilisé pendant vos TP est **PostgreSQL**, qui 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> ou <https://sql.sh/sghd/postgresql>, ainsi que la documentation en français <https://docs.postgresql.fr/> et le site de la communauté française <http://www.postgresql.fr/>. L'outil utilisé comme interface de PostgreSQL est *PgAdmin* (cf. <https://www.pgadmin.org/>).

II. Création de votre base de données en utilisant le SGBD POSTGRESQL installé à Dauphine

Pour créer une base de données depuis les machines du CRIO UNIX de Dauphine, vous devez lancer un navigateur web aller sur le site :

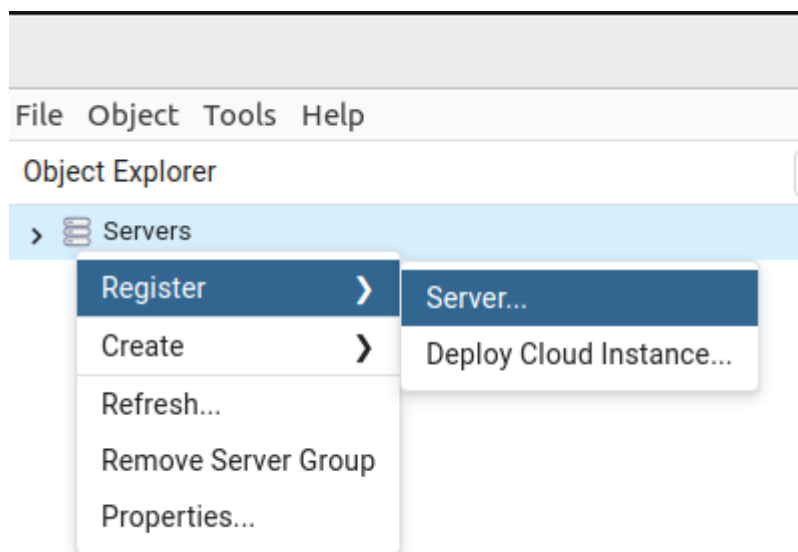
<https://manager.crio.dauphine.fr/mes-projets>, accessible uniquement en local à Dauphine.

Vous devez cliquer sur `creation_projet` et un nom de projet (correspondant à votre nom de base de données et à votre login sous PostgreSQL) ainsi qu'un mot de passe vous seront communiqués. **Veillez à bien noter ces informations.**

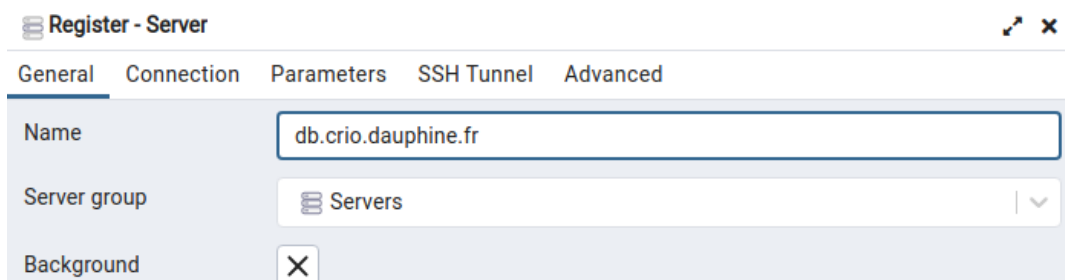
III. Utilisation de POSTGRESQL à Dauphine



1. Lancer *PgAdmin*, l'interface de PostgreSQL.
2. En cliquant avec le bouton droit de la souris sur *Serveur*, sélectionner *Register* et *Server*, comme sur la copie d'écran ci-dessous



3. Une fenêtre va s'afficher. Dans l'onglet *General*, puis dans *Name*, saisir : **db.crio.dauphine.fr**, comme dans la copie d'écran ci-dessous :



Register - Server

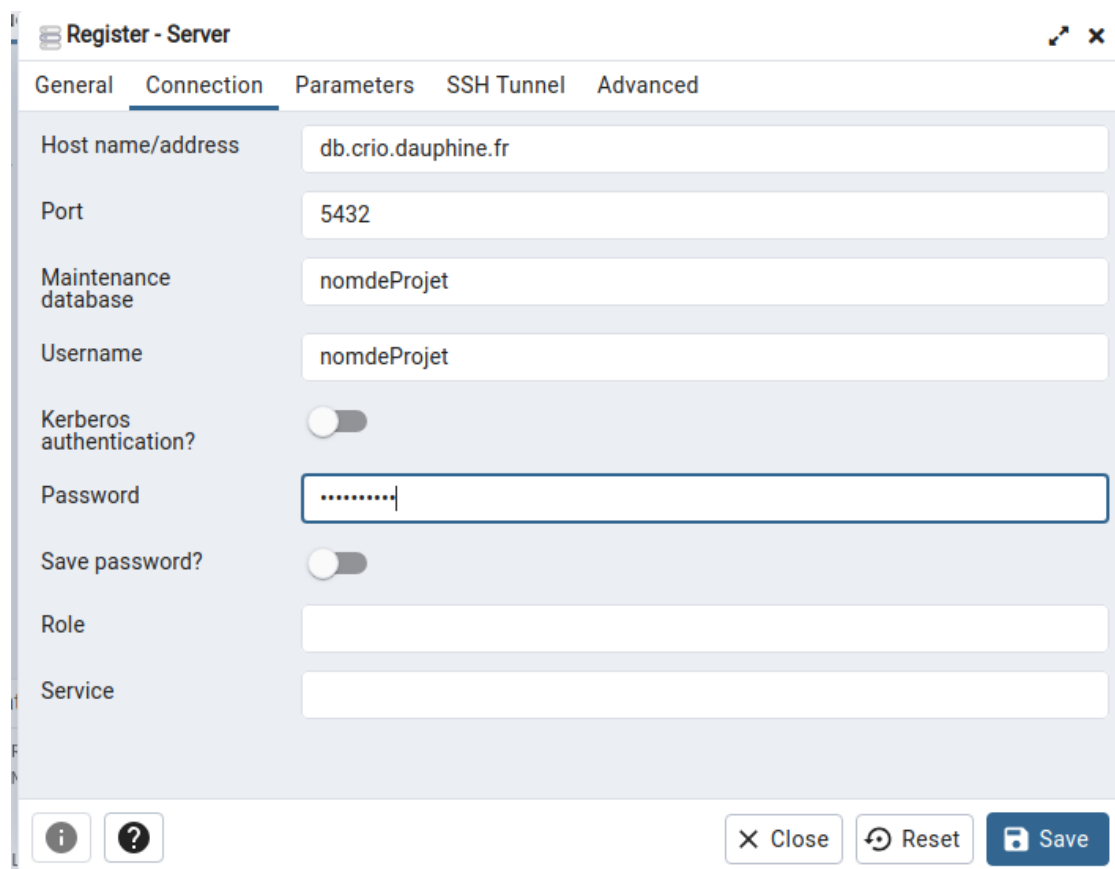
General Connection Parameters SSH Tunnel Advanced

Name db.crio.dauphine.fr

Server group Servers

Background ☐

4. Dans l'onglet *Connection*, saisir **db.crio.dauphine.fr** dans *Host name*, le nom du projet que vous avez créé précédemment (cf. Section II) sur le site <https://manager.crio.dauphine.fr/mes-projets>, dans *Maintenance Base* et dans *Username*, et le mot de passe obtenu précédemment dans *Password*. Puis cliquer sur *Save*, comme indiqué dans la copie d'écran ci-dessous.



Register - Server

General Connection Parameters SSH Tunnel Advanced

Host name/address db.crio.dauphine.fr

Port 5432

Maintenance database nomdeProjet

Username nomdeProjet

Kerberos authentication? ☐

Password

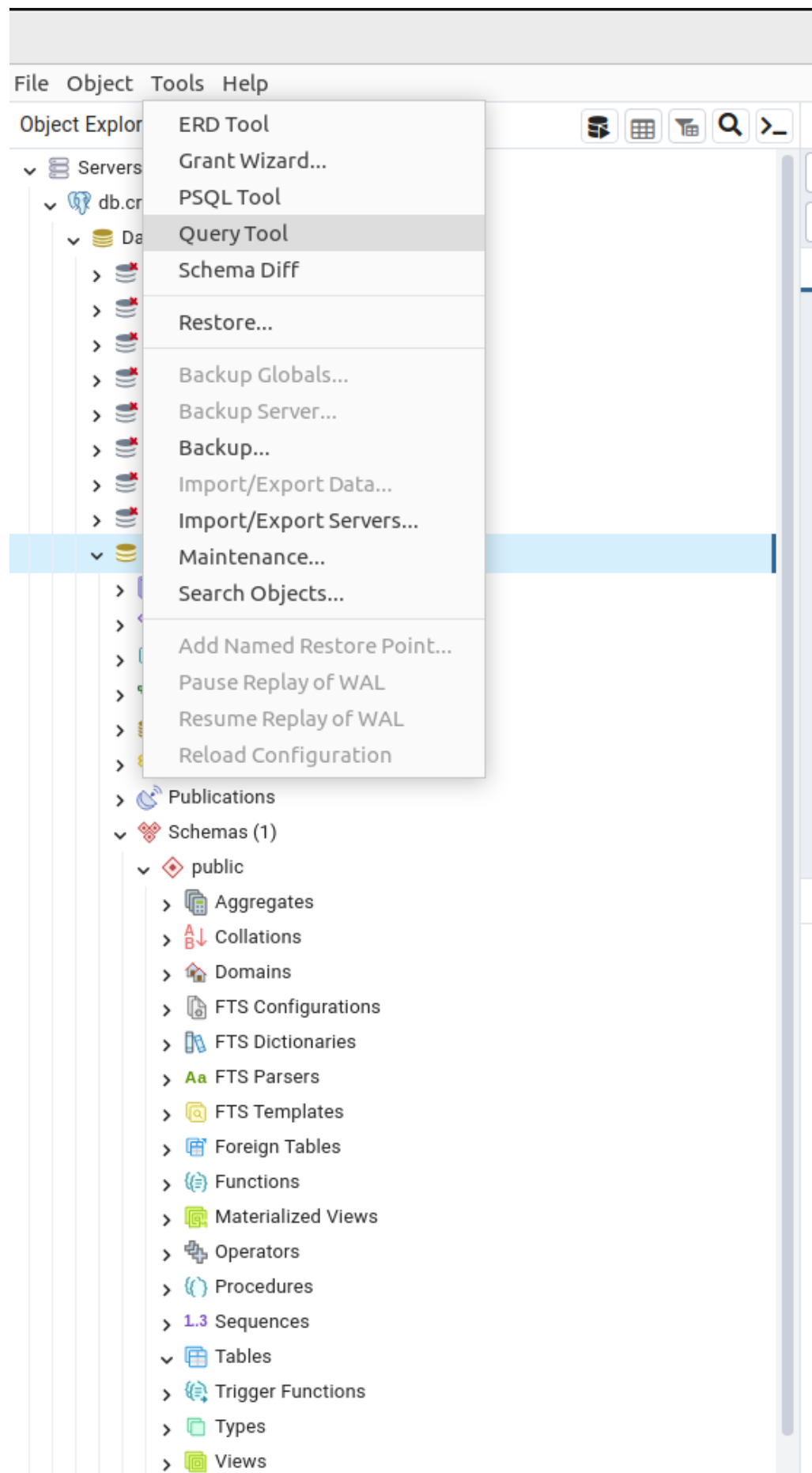
Save password? ☐

Role

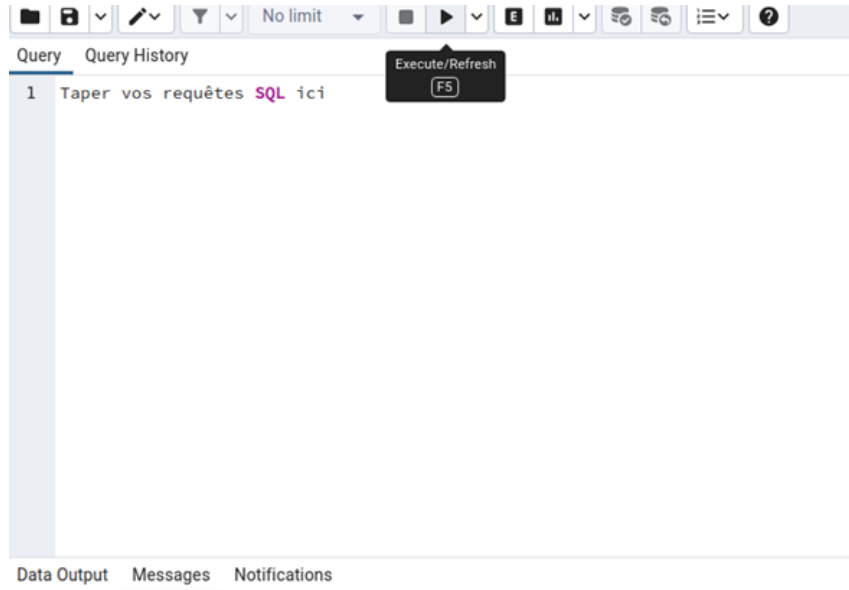
Service

Close Reset Save

5. Cliquer sur le symbole > situé à côté du terme *Servers* dans la fenêtre de gauche, puis sur le symbole > situé à côté de **db.crio.dauphine.fr**, puis sur le symbole > situé à côté de *databases*, puis sur le symbole > correspondant à votre nom de base de données créée à l'étape 1.
6. Dans le Menu *Tools* en haut, cliquer sur *QueryTool* pour lancer l'interpréteur de requêtes SQL (cf. copie d'écran ci-après). Il s'agit du principal outil que vous utiliserez pendant ce TP, l'objectif de ce TP étant le langage SQL.



7. Copier le contenu du scripts SQL de création des tables et d'insertion des nuplets disponibles sur *moodle* (dans la section *SQL*, puis *Script de création de la base de données exemple à utiliser en TP*) dans la fenêtre du *Query Tool* (cf. copie d'écran ci-dessous) et exécutez les requêtes.



8. Le contenu du script SQL est donné dans la section V, les requêtes d'insertion sont rappelées dans la section VI, les fonctions et déclencheurs sont décrits dans la section VII. **L'énoncé du travail à faire est dans la section VIII (page 14).**

Attention : à Dauphine, le schéma *public* est interdit en écriture par défaut. Pour pouvoir créer des relations, il faut créer un nouveau schéma via l'instruction SQL :

CREATE SCHEMA TP ; -- Création d'un schéma nommé TP

Les noms des relations et des fonctions devront tous être précédés du nom du schéma et d'un point ou mettre l'instruction **SET search_path TO TP;** au début du script.

V. SCRIPT DE LA BASE DE DONNEES EXEMPLE

Le script SQL (disponible sur *moodle*) suivant correspond aux commandes de création de la base de données exemple sur laquelle porte le TP.

La base de données contient 6 relations (tables) permettant de gérer des départements, des étudiants, des cours, des enseignants et des réservations de salles pour ces cours et ces enseignements.

Attention : à Dauphine, il faut ajouter l'instruction **CREATE SCHEMA TP ;** pour créer le schéma nommé TP) puis l'instruction **SET search_path TO TP;** au début du script, pour se positionner sur le schéma TP.

Dans la suite du document, le script correspond au script du schéma public (sans le TP.).

```
CREATE TABLE Departement
(
  Departement_id      serial,
  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      serial,
  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 Cours

```
(
  Cours_ID integer NOT NULL,
  Departement_ID integer NOT NULL,
  Intitule   varchar(60) NOT NULL UNIQUE,
  Description varchar(1000),
  -- Attention : un cours est identifié par 2 attributs
  CONSTRAINT PK_Cours PRIMARY KEY (Cours_ID, Departement_ID),
  CONSTRAINT "PK_Cours_Departement"
    FOREIGN KEY (Departement_ID)
      REFERENCES Departement (Departement_ID)
      ON UPDATE RESTRICT ON DELETE RESTRICT
);
```

CREATE TABLE Enseignant

```
(
  Enseignant_ID      serial,
  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),
  -- Une salle est identifiée par 2 attributs
  CONSTRAINT PK_Salle PRIMARY KEY (Batiment, Numero_Salle)
);
```



```

CREATE TABLE Reservation
(
    Reservation_ID          serial,
    Batiment                varchar(1) NOT NULL,
    Numero_Salle            varchar(10) NOT NULL,
    Cours_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_Cours"
        FOREIGN KEY (Cours_ID,Departement_ID)
REFERENCES Cours (Cours_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)
);

-- Un exemple de vue
CREATE OR REPLACE VIEW Email_Etudiant
AS SELECT Nom, Prenom, Email FROM Etudiant;

```

Remarque : Dans le script précédent, on a choisi d'utiliser des clés artificielles (de type SERIAL) pour toutes les relations, sauf Salle et Cours. En effet, on a choisi pour Salle, une clé métier car une salle est identifiée par un couple (Batiment, Numero_Salle), qui ne peut pas être généré artificiellement par le SGBD. De même, un cours est identifié par le couple (Cours_ID, Departement_ID), car on a choisi de numéroter les cours par département (on a le cours 1 du département 1, le cours 1 du département 2 etc.).

Sous PostgreSQL, l'utilisation du type SERIAL implique la création implicite d'une séquence qui va permettre au SGBD de générer les valeurs de la clé primaire. Quand vous avez créé la relation Departement, par exemple, vous avez dû obtenir le message suivant :

```

=>NOTICE:CREATE TABLE will create implicit sequence "
departement_departement_id_seq" for serial column "
departement_departement_id"

```

Indiquant que pour la relation Departement, le SGBD a créé une séquence nommée departement_departement_id_seq.

Vous pouvez faire une requête pour avoir les informations de la séquence :

```
SELECT * FROM departement_departement_id_seq ;
```

Une insertion dans la relation Departement se fait soit en ne donnant que la valeur de l'attribut Nom_Departement :

```
INSERT INTO Departement(Nom_Departement) VALUES ('MIDO');
```

soit en faisant explicitement appel à la séquence :

```
INSERT INTO Departement
VALUES (nextval('departement_departement_id_seq'),'DEP');
```

où `nextval('departement_departement_id_seq')` permet d'obtenir la prochaine valeur dans la séquence.

VI. INSERTION DES NUPLETS DE LA BASE EXEMPLE

Les commandes SQL permettent d'insérer des nuplets dans la base de données.

```
INSERT INTO Departement(Nom_Departement) VALUES ('MIDO');
INSERT INTO Departement(Nom_Departement) VALUES ('LSO');
INSERT INTO Departement(Nom_Departement) VALUES ('MSO');
```



```
INSERT INTO
Etudiant(Nom,Prenom,Date_Naissance,Adresse,Ville,Code_Postal,Telephone,Fax,
Email) VALUES ('GAMOTTE', 'Albert','1979/02/18','50, Rue des
alouettes','PARIS','75021','0143567890',NULL, 'gamotal4[at]etud.dauphine.fr'
);
INSERT INTO
Etudiant(Nom,Prenom,Date_Naissance,Adresse,Ville,Code_Postal,Telephone,Fax,
Email) VALUES ('HIBULAIRE', 'Pat','1980/08/23','10, Avenue des
marguerites','POUILLOIN','40000','0678567801',NULL, 'pat[at]yahoo.fr');
INSERT INTO
Etudiant(Nom,Prenom,Date_Naissance,Adresse,Ville,Code_Postal,Telephone,Fax,
Email) VALUES ('ODENT', 'Jamal','1978/05/12','25, Boulevard des
fleurs','PARIS','75022','0145678956','0145678956', 'odent[at]free.fr');
INSERT INTO
Etudiant(Nom,Prenom,Date_Naissance,Adresse,Ville,Code_Postal,Telephone,Fax,
Email) VALUES ('DEBECE', 'Gill','1979/07/15','56, Boulevard des
fleurs','PARIS','75022','0678905645',NULL, 'deby[at]hotmail.com');
INSERT INTO
Etudiant(Nom,Prenom,Date_Naissance,Adresse,Ville,Code_Postal,Telephone,Fax,
Email) VALUES ('DEBECE', 'Aude','1979/08/15','45, Avenue des
abeilles','PARIS','75022',NULL,NULL,NULL);
```



```
INSERT INTO Enseignant(Departement_ID,Nom,Prenom,Grade,Telephone,Fax,Email)
VALUES ((SELECT Departement_id FROM Departement WHERE
nom_departement='MIDO')
, 'MANOUVRIER', 'Maude', 'MCF', '4185', '4091', 'maude.manouvrier[at]dauphine.fr'
);
INSERT INTO Enseignant
(Departement_ID,Nom,Prenom,Grade,Telephone,Fax,Email)
VALUES (1, 'BELHAJJAME', 'Khalid
', 'MCF', NULL, NULL, 'khalid.belhajjame[at]dauphine.fr');
INSERT INTO Enseignant(Departement_ID,Nom,Prenom,Grade,Telephone,Fax,Email)
VALUES (1, 'NEGRE', 'Elsa ', 'MCF', NULL, NULL, 'elsa.negre[at]dauphine.fr');
INSERT INTO Enseignant
(Departement_ID,Nom,Prenom,Grade,Telephone,Fax,Email)
VALUES (1, 'MURAT', 'Cecile ', 'MCF', NULL, NULL, 'cecile.murat[at]dauphine.fr');
```



```
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 Cours VALUES ('1','1','Bases de Données Relationnelles','Niveau
Licence (L3) : Modélisation E/A et UML, Modèle relationnel, Algèbre
Relationnelle, Calcul relationnel, SQL, dépendances fonctionnelles et formes
normales');
INSERT INTO Cours VALUES ('2','1','Langage C++','Niveau Master 1');
INSERT INTO Cours VALUES ('3','1','Mise à Niveau Bases de Données','Niveau
Master 2 - Programme Licence et Master 1 en Bases de Données');

```

```

INSERT INTO
Reservation(Batiment,Numero_Salle,Cours_ID,Departement_ID,Enseignant_ID,Dat
e_Resa,Heure_Debut,Heure_Fin,Nombre_Heures) VALUES
('B','022','1','1',(SELECT Enseignant_id FROM Enseignant WHERE
Nom='MANOUVRIER'),'2016/10/15','08:30:00','11:45:00','3');
INSERT INTO
Reservation(Batiment,Numero_Salle,Cours_ID,Departement_ID,Enseignant_ID,Dat
e_Resa,Heure_Debut,Heure_Fin,Nombre_Heures) VALUES
('B','022','1','1','2','2016/11/04','08:30:00','11:45:00','3');
INSERT INTO
Reservation(Batiment,Numero_Salle,Cours_ID,Departement_ID,Enseignant_ID,Dat
e_Resa,Heure_Debut,Heure_Fin,Nombre_Heures) VALUES
('B','022','1','1','2',DEFAULT,'08:30:00','11:45:00','3');
INSERT INTO
Reservation(Batiment,Numero_Salle,Cours_ID,Departement_ID,Enseignant_ID,Dat
e_Resa,Heure_Debut,Heure_Fin,Nombre_Heures) VALUES
('B','020','1','1','1',DEFAULT,'08:30:00','11:45:00','3');

```

VII. EXEMPLE DE FONCTIONS SQL, PL/PGSQL ET DE DECLENCHEUR SOUS POSTGRESQL

Sous PostgreSQL, vous pouvez créer des fonctions SQL, correspondant à des requêtes paramétrées et des déclencheurs. Les sous-sections suivantes vous présentent des exemples.

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'));
```

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.

```
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 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;  ← 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)

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
        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
                      en cours d'insertion, représenté par NEW
    END IF;
END; '
LANGUAGE 'plpgsql';
```

La fonction `FunctionTriggerReservation()` est utilisée dans un déclencheur (ou *trigger* en anglais – voir 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.

VIII. TRAVAIL A FAIRE PENDANT LE TP

A. Création de la base exemple

1. Créer le schéma de données de la base exemple en exécutant le script dans l'interpréteur de requêtes SQL.

Il vous suffit pour cela de recopier le contenu du fichier dans la partie gauche de la fenêtre de dbfiddle (voir Figure 5).

2. 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 cours existant,
 - f. Une réservation qui chevauche une réservation existante (pour tester l'affichage du déclencheur).

Rappel : sous PostgreSQL, pour mettre des lignes en commentaires, vous devez précéder chaque ligne par – (deux tirets) ou placer les lignes à mettre en commentaire entre `/*` et `*/`.

B. Interrogation de la base de données exemple

Ecrire et exécuter les requêtes¹ d'interrogation SQL suivantes sur la base de données exemple (sous *QueryTool* si vous utilisez le SGBD PostgreSQL installé à Dauphine ou dans la partie droite de la fenêtre si vous utilisez DBFiddle et en cliquant sur Run) :

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 enseignants n'ayant pas de Fax (valeur **NULL**) ?
8. Quels sont les intitulés des cours dont la description contient le mot 'SQL' ou 'Licence' ?
9. Si on suppose qu'une heure de cours coûte 50 euros, quel est le coût en euros de chaque cours (les heures de cours concernent les heures réservées – voir relation *Réservation*)?
10. A partir de la requête précédente, indiquer quels sont les intitulés des cours dont le coût est compris entre 500 et 750 euros.

¹ **Attention** : toutes les requêtes ne retournent pas forcément 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.

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 de cours associé 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 cours 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 cours (un étudiant peut avoir plusieurs notes pour le même cours).
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 ce cours (sinon ajouter l'inscription de l'étudiant au cours).

IV. Description de DBFIDDLE

Si vous ne souhaitez pas utiliser le SGBD installé à Dauphine et si vous ne souhaitez pas installer PostgreSQL sur votre machine, vous pouvez utiliser un site en ligne gratuit : [**https://www.db-fiddle.com/**](https://www.db-fiddle.com/)

Ce site web permet de créer en ligne une base de données et de l'interroger en SQL, sans avoir à installer de SGBD sur votre machine.

Attention : la connexion est temporaire, votre base est effacée une fois que vous quittez le site.

Dans la partie gauche de la fenêtre de dbfiddle (voir Figure 1), vous devez y saisir/copier le script SQL de création du schéma, ainsi que les commandes d'insertion

et de mises à jour des nuplets (i.e. toutes les commandes CREATE, INSERT, UPDATE et DELETE).

Pour créer la base de données (temporaire – valide uniquement le temps de votre connexion sur le site), il faut **bien sélectionner PostgreSQL** dans la liste déroulante de droite des SGBD disponibles, puis il faut cliquer sur **Run**.

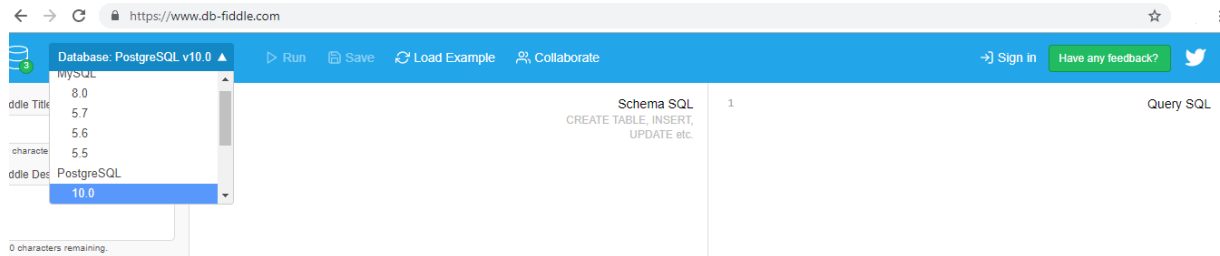


Figure 1 - Interface de DBFiddle - à gauche l'espace pour les requêtes de création de tables, les insertions et les déclencheurs - à droite les requêtes d'interrogation.

Vos requêtes d'interrogation (**SELECT**) peuvent être saisies dans la partie droite de la fenêtre. L'exécution se fait en cliquant sur **Run**. Le résultat de la requête apparaît en bas de la fenêtre (voir Figure 2).

Results Copy as Markdown

Query #1 Execution time: 1ms

enseignant_id	departement_id	nom	prenom	grade	telephone	fax	email
1	1	MANOUVRIER	Maude	MCF	4185	4091	maude.manouvrier@dauphine.fr
2	1	BELHAJJAME	Khalid	MCF			khalid.belhajjame[at]dauphine.fr
3	1	NEGRE	Elsa	MCF			elsa.negre[at]dauphine.fr
4	1	MURAT	Cecile	MCF			cecile.murat[at]dauphine.fr

Figure 2 - Exemple d'exécution de requêtes sous DBFiddle.

Vous pouvez également tester les déclencheurs en plaçant bien la requête qui doit lancer votre déclencheur après la définition de ce dernier, dans la fenêtre de gauche et en cliquant sur **Run**.

Results

Schema Error: error: Reservation impossible, salle occupée à la date et aux horaires demandés

Figure 3 - Exemple d'exécution d'un déclencheur sous DBFiddle (déclencheur défini en cours et manipulé pendant le TP).

V. Description d'un site *bac à sable* permettant d'utiliser PostgreSQL

Si vous ne souhaitez pas utiliser le SGBD installé à Dauphine et si vous ne souhaitez pas installer PostgreSQL sur votre machine, vous pouvez utiliser également le site en ligne gratuit : <https://extendsclass.com/postgresql-online.html> - cf copie d'écran sur la Figure 4.

Contrairement au site *dbfiddle* décrit dans la section IV, ce site ne recrée pas la base à chaque fois. La base de données est conservée le temps de votre connexion sur le site.

Vous devez donc exécuter une première fois le script de création de votre base de données, puis effacer le script et saisir vos requêtes. Attention, le site limitant la taille des requêtes, il faut exécuter le script par petits bouts (d'abord la création des relations, puis la création des fonctions et déclencheurs, puis les *inserts*).

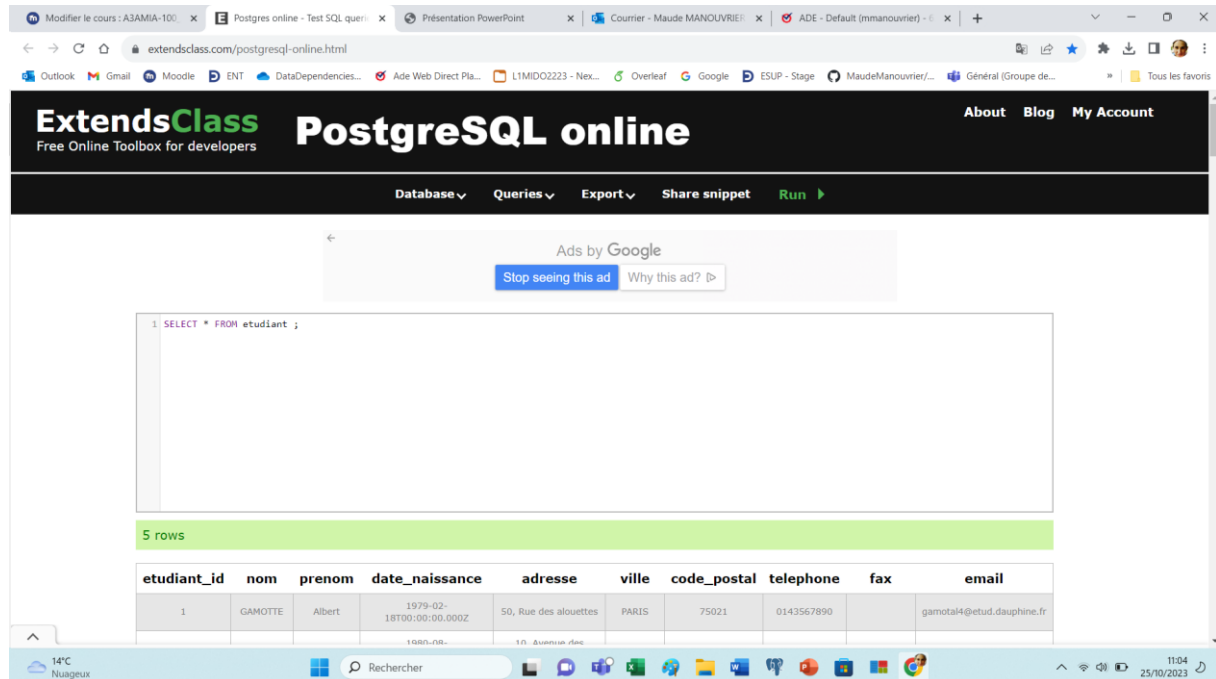


Figure 4 - Interface du site <https://extendsclass.com/postgresql-online.html>

IX. ANNEXE : INSTALLATION DE POSTGRESQL SUR VOTRE MACHINE

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 :

- <https://www.postgresql.org/>
- <https://sql.sh/sghd/postgresql>
- Documentation en français : <https://docs.postgresql.fr/>
- Site de la communauté française : <https://www.postgresql.fr/>

Pour installer PostgreSQL sur votre machine, vous pouvez regarder les liens suivants :

- <https://www.postgresqltutorial.com/install-postgresql/>
- <https://www.postgresql.fr/media/doc postgresql 9 1.pdf>
- <https://docs.postgresql.fr/12/INSTALL.html>

Vous pouvez télécharger le SGBD sur : <https://www.postgresql.org/download/>

Une interface graphique pour utiliser PostgreSQL est PgAdmin, téléchargeable à l'adresse : <https://www.pgadmin.org/>

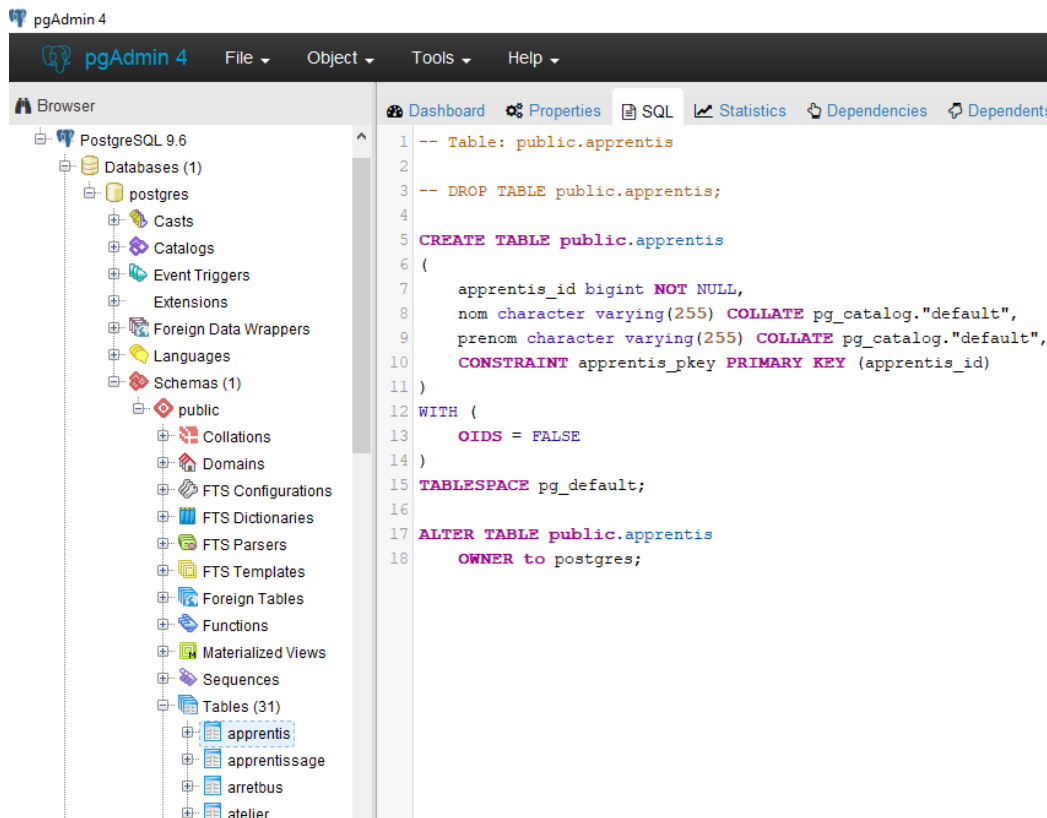


Figure 5 - Interface de PgAdmin 4

X. ANNEXE : EXEMPLES DE PROGRAMMES JDBC

JDBC est un *middelware* facilitant la connexion entre un client de bases de données 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**, 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** (ou *driver*) particulier. Ce pilote permet la traduction des commandes JDBC en commandes spécifiques au SGBD utilisé.

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 est la suivante : `jdbc:postgresql://url_du_SGBD/nom_base`