Module SLAM 3 TP3

Propriétés	Description
Intitulé	**INIT_POSTGRESQL**
Outils	• A.G.L WIN'DESIGN • SGBD POSTGRESQL
Durée estimée en heures	4 Heures
Savoir-faire module SLAM3	• Concevoir une base de données • Valider un schéma de base de données • Programmer dans l'environnement de développement associé à un SGBD
Savoirs Module SLAM3	• Modèles de représentation des données • Langage de programmation associé à un SGBD
Documents joints	Fiche d'exploitation pédagogique Annexes
Réception	Mise en place et exploitation du SGBD Postgresql
Equipe	Seul × Par équipe de ... □ 2□ 3 □ 4

Préalable : Lire le document de présentation rapide de POSTGRESQL

Sites de référence : **PostgreSQL : http://www.postgresql.org**
Version française : http://www.postgresql.fr

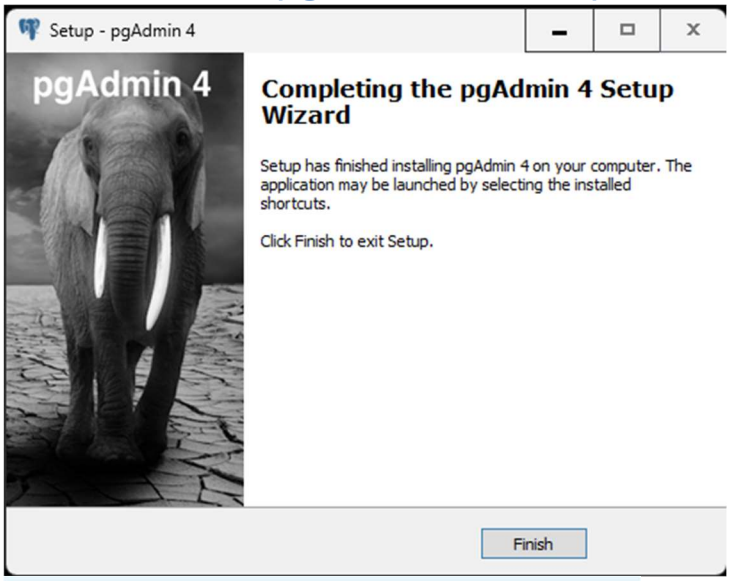
Première partie : Utilisation de POSTGRESQL en local

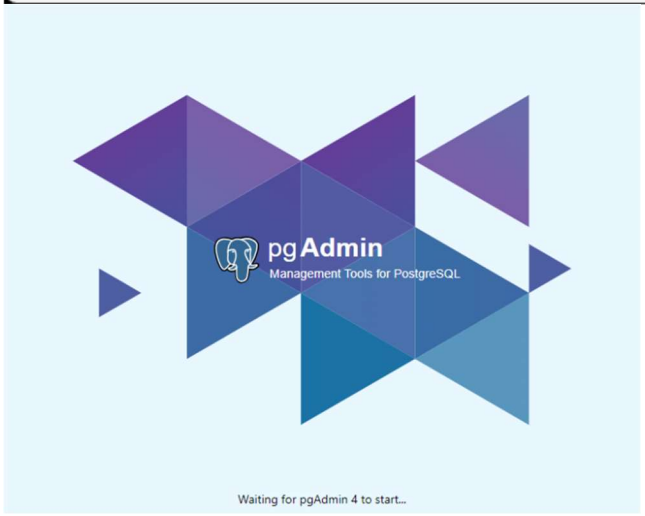
Environnement matériel : Utiliser la machine hôte WINDOWS mise à votre disposition par le professeur de SLAM3.

Etape 1 - Installer PostgreSQL pour Windows

Vous pouvez trouver plusieurs types d'installeurs pour Windows.Les plus usités sont créés par des entreprises (BigSql, EnterpriseDB, …) d'autres sont créés par la communauté.
L'installeur "en un clic" **(One Click Installer)** 'est la méthode recommandée pour une première prise de contact avec PostgreSQL, car elle se charge d'effectuer les opérations de configuration du serveur après l'installation proprement dite. Elle se charge également d'installer tous les composants et outils utiles pour administrer et utiliser PostgreSQL.

Installation de pgAdmin 4 sur le post de travail





Etape 2 - Tester la connexion à PostgreSQL avec les outils d'administration

A faire :

Vérifiez la présence de **PostgreSQL** dans la liste des processus du gestionnaire des taches

Les outils d'administration :

- ***PSQL***

Psql est une interface en ligne de commande permettant d'accéder à PostgreSQL
Documentation en ligne : ***http://docs.postgresqlfr.org***

A faire : Démarrer **Psql** (application **psql** dans l'interface W10)
Tester la connexion avec l'utilisateur ***postgres***
Tester quelques méta-commandes trouvées sur la documentation comme :

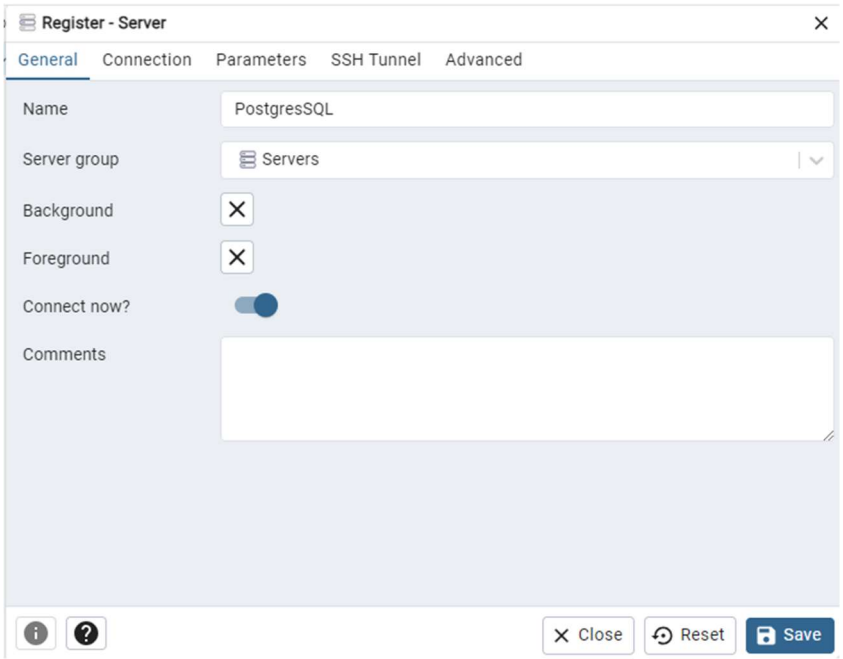
Afficher l'aide	**help ou \ ?**
Afficher la liste des bases de données	**\l**
Afficher la liste des tables	**\dt**
Afficher la liste des utilisateurs	**\du**
Afficher la liste des langages installés	**\dL**
Quitter **psql**	

- ***PGADMIN***

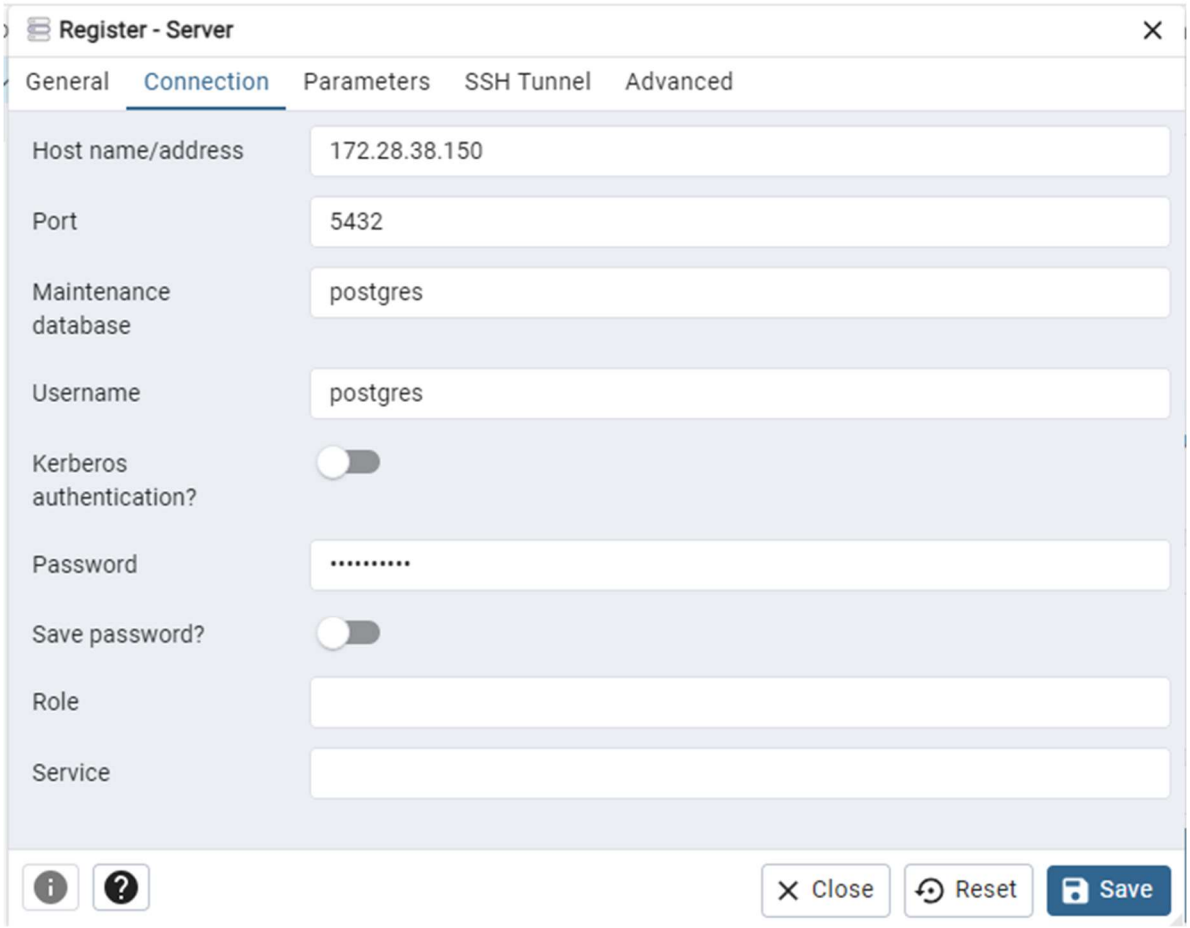
C'est un outil d'administration graphique pour PostgreSQL
Documentation en ligne : ***http://www.pgadmin.org/***
A faire :

Tester la connexion à la base ***postgres*** et manipuler l'outil (base, schéma, rôle)

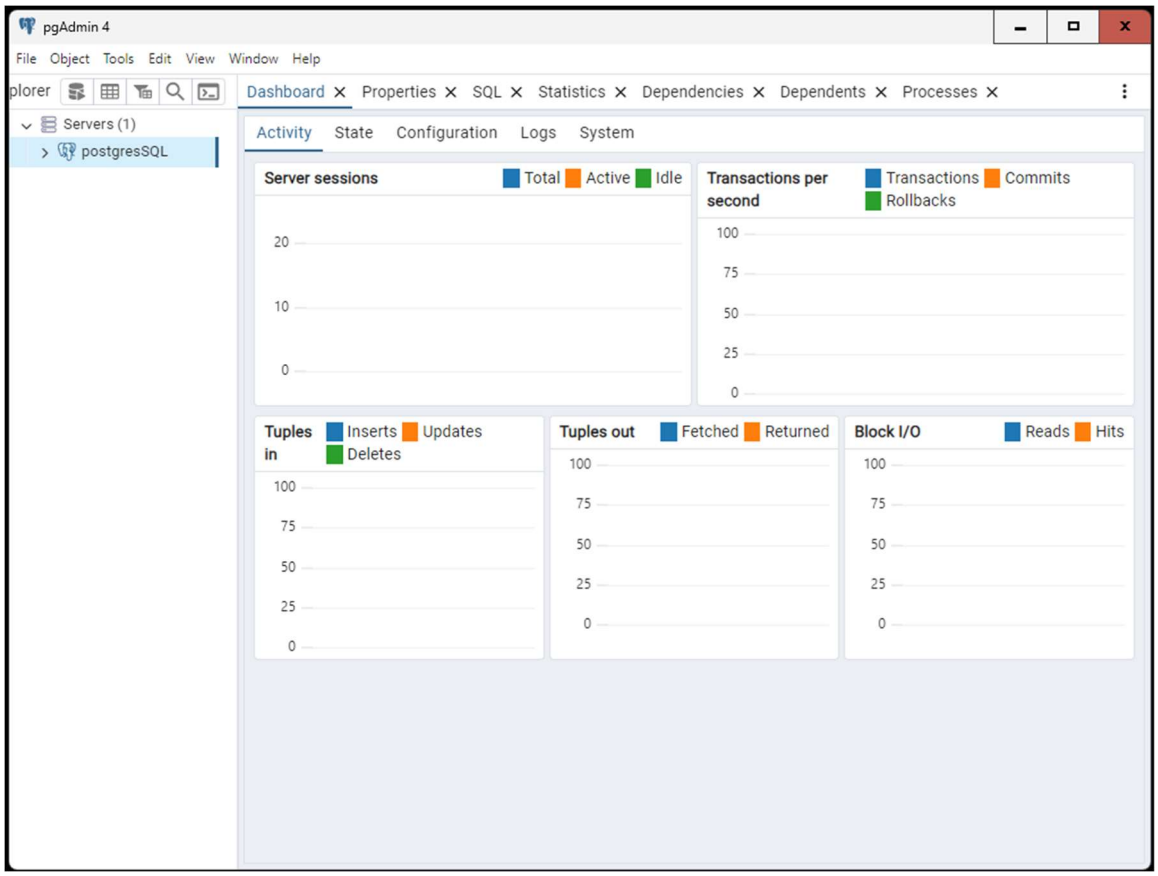
Connexion à la base postgres en attribuant un nom au serveur de connexion



Connexion à la base à l'adresse 172.28.38.150 avec le mot de passe Btssio82300



La connexion a bien était faite



Etape 3 - Créer une base de données PostgreSQL

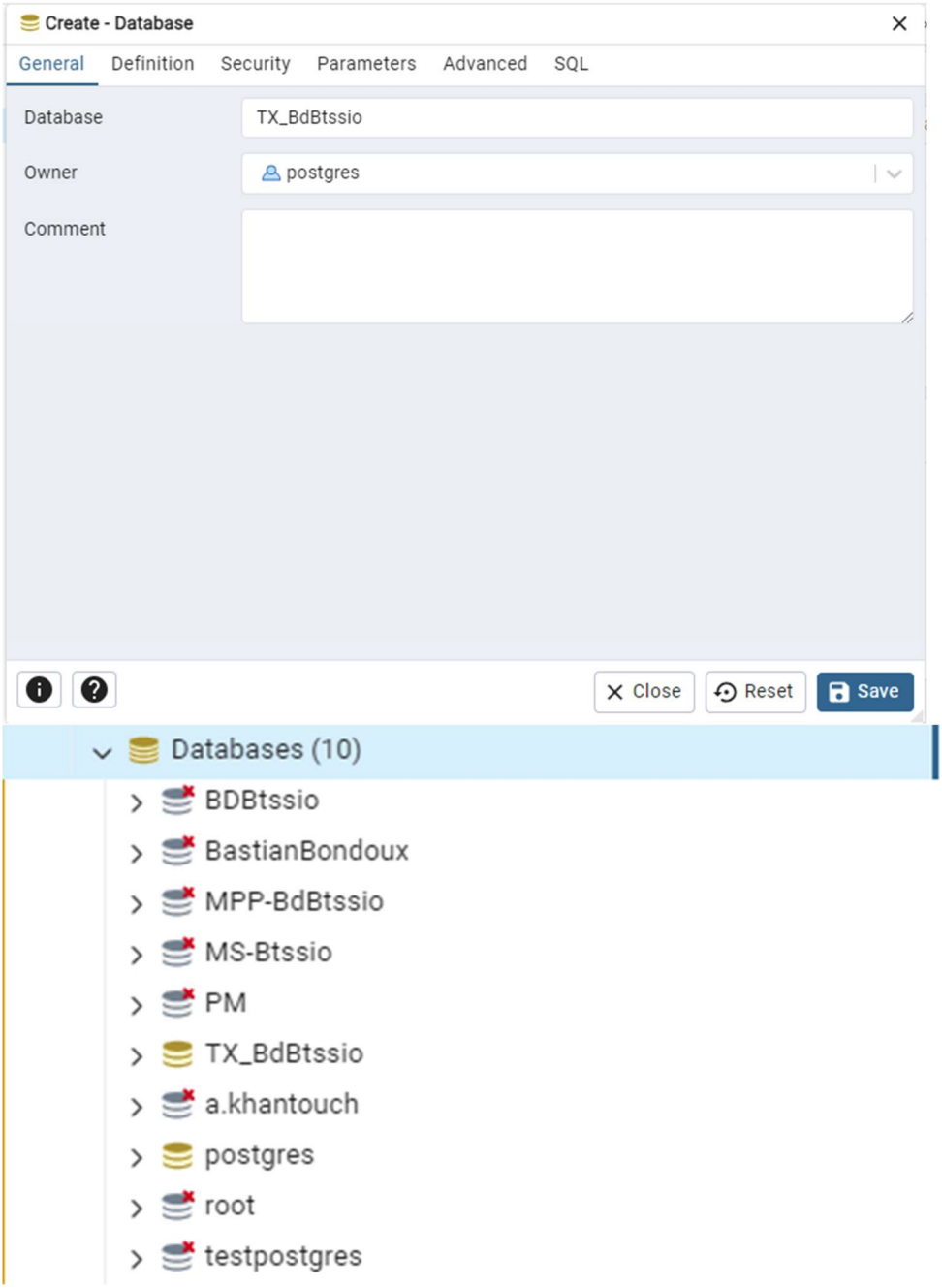
La base de données concerne une partie de la structure des enseignements en BTSSIO.
Le schéma des données a été réalisé avec WINDESIGN.

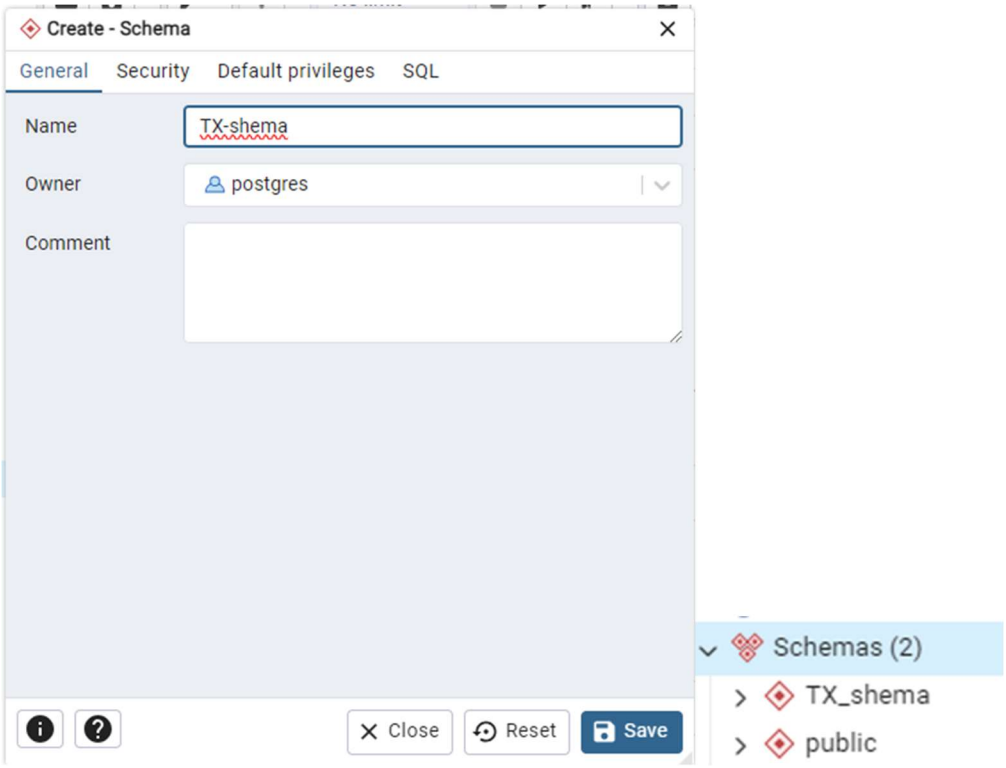
A faire :

- Récupérer le fichier .mcd correspondant appelé **BdBtssioMCD**
- Avec **WINDESIGN :** générer le script pour ***Postgresql***
- Avec **PgAdmin :** créer la base de données **BdBtssio**
  exécuter le script dans l'éditeur SQL

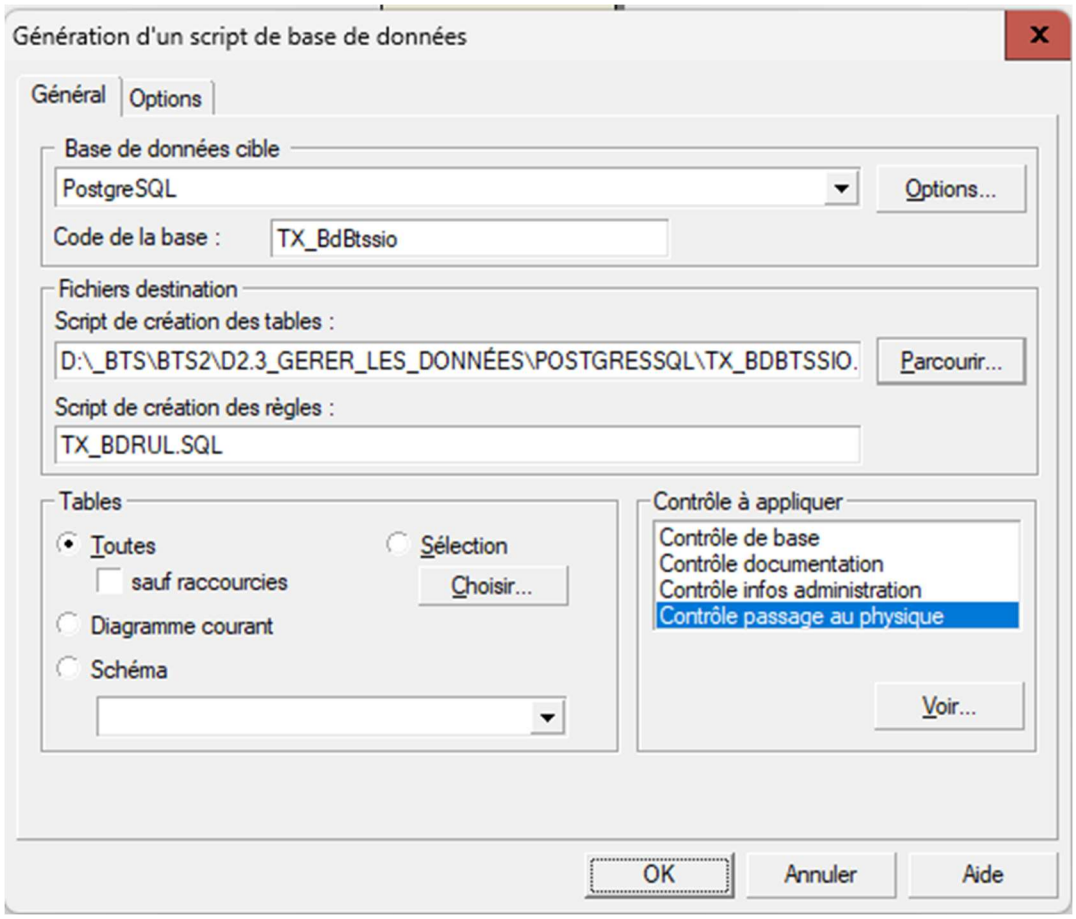
☐ Vérifier la création effective des tables dans le schéma de la base de données.

Création d'une base de données comportant pour nom TX_BdBtssio

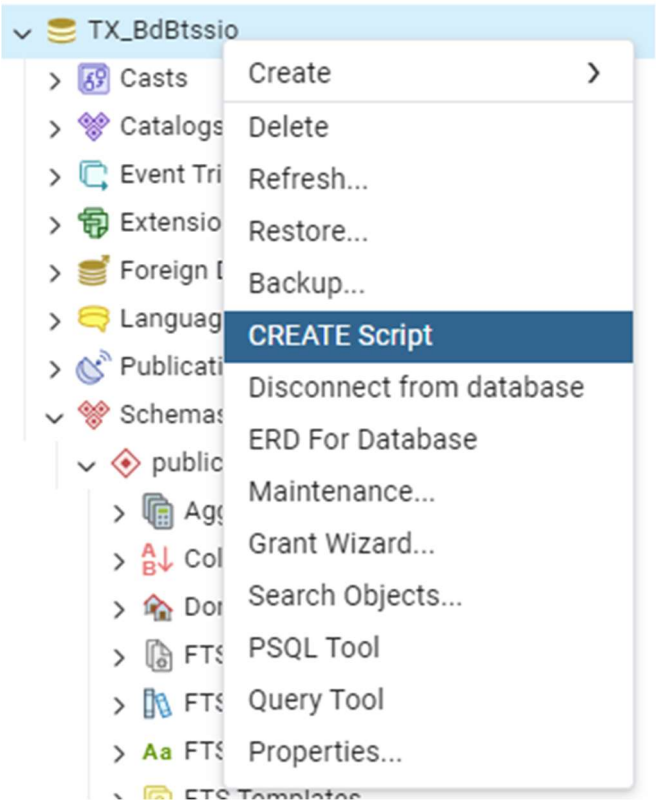




Création du nouveau schéma



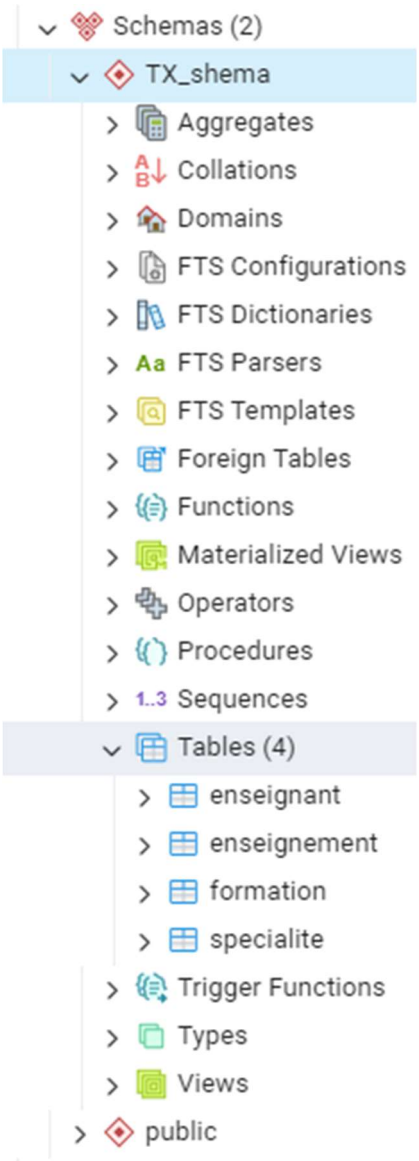
Création du script SQL pour créer les tables de la base de données



Insertion des tables dans la base de données

Query | Query History

```
SET schema 'TX_shema';
-- ----------------------------------------------------------------------------
--       TABLE : ENSEIGNEMENT
-- ----------------------------------------------------------------------------

CREATE TABLE ENSEIGNEMENT
   (
    SPECIALITE_ID int4 NOT NULL  ,
    ENSEIGNEMENT_ID int4 NOT NULL  ,
    INTITULE varchar(60) NOT NULL  ,
    DESCRIPTION varchar(1000) NULL
,   CONSTRAINT PK_ENSEIGNEMENT PRIMARY KEY (SPECIALITE_ID, ENSEIGNEMENT_ID)
   ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE ENSEIGNEMENT
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_ENSEIGNEMENT_SPECIALITE
     ON ENSEIGNEMENT (SPECIALITE_ID)
    ;

-- ----------------------------------------------------------------------------
--       TABLE : SPECIALITE
-- ----------------------------------------------------------------------------

CREATE TABLE SPECIALITE
   (
    SPECIALITE_ID int4 NOT NULL  ,
    FORMATION_ID int4 NOT NULL  ,
    ENSEIGNANT_ID int4 NOT NULL  ,
    NOM_SPECIALITE varchar(25) NOT NULL
,   CONSTRAINT PK_SPECIALITE PRIMARY KEY (SPECIALITE_ID)
   ) ;

```



Les tables ont bien était inséré à la base de données

Etape 4 - Interroger la base de données PostgreSQL

La base de données va être remplie afin de pouvoir réaliser quelques traitements

A faire :
Lancer l'insertion des n-uplets en exécutant le script d'insertion **BdBtssioInsertion.sql.**

Analyser le message d'erreur et modifier le script d'insertion en conséquence.

Il fait intervertir les valeurs dans l'insertion enseignant car la première colonne correspond au ID de la spécialité et la deuxième colonne correspond donc à l'ID de la spécialité
Il faut tout aussi définir les ID de 1 à 4

TX_BdBtssio/postgres@postgresSQL

No limit

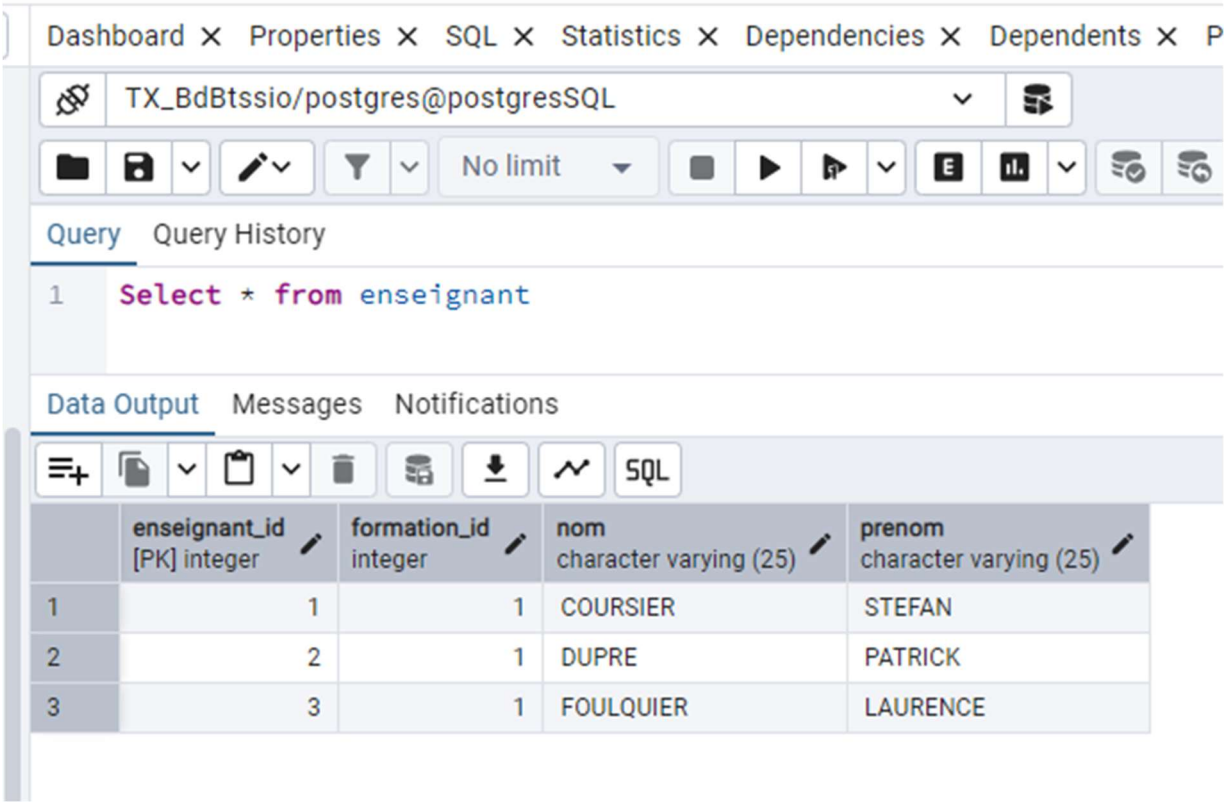
Query Query History

```
1  set schema 'TX_shema';
2
3  INSERT INTO formation VALUES ('1','BTS SIO');
4  INSERT INTO formation VALUES ('2','BTS CGO');
5
6  INSERT INTO enseignant VALUES('1',(SELECT formation_id FROM formation WHERE nom_formation='BTS SIO'),'COURSIER','STEFAN');
7  INSERT INTO enseignant VALUES('2',(SELECT formation_id FROM formation WHERE nom_formation='BTS SIO'),'DUPRE','PATRICK');
8  INSERT INTO enseignant VALUES('3',(SELECT formation_id FROM formation WHERE nom_formation='BTS SIO'),'FOULQUIER','LAURENCE');
9
10 INSERT INTO specialite VALUES ('1','1',(SELECT enseignant_id FROM enseignant WHERE nom='COURSIER'),'SLAM');
11 INSERT INTO specialite VALUES ('2','1',(SELECT enseignant_id FROM enseignant WHERE nom='FOULQUIER'),'SISR');
12
13 INSERT INTO enseignement VALUES ((SELECT specialite_id FROM specialite WHERE nom_specialite='SLAM'),'1','SGBD','Bases de données');
14 INSERT INTO enseignement VALUES ((SELECT specialite_id FROM specialite WHERE nom_specialite='SLAM'),'2','PROG','Programmation');
15 INSERT INTO enseignement VALUES ((SELECT specialite_id FROM specialite WHERE nom_specialite='SISR'),'3','SERVICES','Exploitation des services');
16 INSERT INTO enseignement VALUES ((SELECT specialite_id FROM specialite WHERE nom_specialite='SISR'),'4','ADMIN','Administration des systemes');
```

A faire :

Réaliser les requêtes suivantes avec **Psql et/ou PgAdmin**

- Liste des enseignants (nom et prénom)



	enseignant_id [PK] integer	formation_id integer	nom character varying (25)	prenom character varying (25)
1	1	1	COURSIER	STEFAN
2	2	1	DUPRE	PATRICK
3	3	1	FOULQUIER	LAURENCE

- Liste des enseignements de la spécialité SLAM (intitulé et description)

Query Query History

```
1 select specialite.nom_specialite, enseignement.intitule, enseignement.description from enseignement
2 natural join specialite
3 where specialite.nom_specialite = 'SLAM';
```

Data Output Messages Notifications

	nom_specialite character varying (25)	intitule character varying (60)	description character varying (1000)
1	SLAM	SGBD	Bases de données
2	SLAM	PROG	Programmation

- Liste des enseignants responsables par spécialité (1seul par spécialité) (nom_spécialité, nom, prénom)

Dashboard Properties SQL Statistics Dependencies Dependents Processes TX_BdBtssio/postgres@postgresSQL*

TX_BdBtssio/postgres@postgresSQL — No limit

Query Query History

```
1 select specialite.nom_specialite, enseignant.nom, enseignant.prenom from specialite
2 natural join enseignant
3
```

Data Output Messages Notifications

	nom_specialite character varying (25)	nom character varying (25)	prenom character varying (25)
1	SLAM	COURSIER	STEFAN
2	SISR	FOULQUIER	LAURENCE

- Nombre d'enseignements par spécialité (nom_spécialité, nombre)

TX_BdBtssio/postgres@postgresSQL — No limit

Query Query History

```
1 select specialite.nom_specialite, count(enseignant.enseignant_id) from specialite
2 natural join enseignant
3 group by specialite.nom_specialite;
```

Data Output Messages Notifications

	nom_specialite character varying (25)	count bigint
1	SISR	1
2	SLAM	1

Etape 5 - Modifier le schéma de la base de données PostgreSQL

Dans un premier temps, il s'agira de compléter la base de données par la création de tables supplémentaires assurant la gestion des étudiants.
Pour cela vous disposez du script **BdBtssioCreationEtudiant.sql**.

A faire :
Lancer l'exécution du script **BdBtssioCreationEtudiant.sql** dans PgAdmin
Insérer manuellement les occurrences suivantes d'étudiants avec PgAdmin , ou avec Psql

Query Query History

```
1  CREATE TABLE Etudiant
2  (
3   Etudiant_ID    integer NOT NULL,
4   Nom        varchar(25) NOT NULL,
5   Prenom     Varchar(25) NOT NULL,
6   Email          Varchar(100) DEFAULT NULL,
7   CONSTRAINT PK_Etudiant PRIMARY KEY (Etudiant_ID) -- D?finition de la cl? primaire
8  );
9
10 CREATE TABLE Inscription
11 (
12  Etudiant_ID     integer NOT NULL,
13  Enseignement_ID integer NOT NULL,
14  Specialite_ID integer NOT NULL,
15  CONSTRAINT PK_Inscription PRIMARY KEY (Etudiant_ID,Enseignement_ID,Specialite_ID),
16  CONSTRAINT "FK_Inscription_Etudiant" FOREIGN KEY (Etudiant_ID) REFERENCES Etudiant
17  CONSTRAINT "FK_Inscription_Enseignement" FOREIGN KEY (Enseignement_ID,Specialite_I
18  CONSTRAINT "UN_Inscription" UNIQUE (Etudiant_ID,Enseignement_ID,Specialite_ID) --
19 );
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 77 msec.
```

etudiant_id [PK] integer	nom character var	prenom character var	email character varying(100)
1	debece	aude	aude.debece@gmail.com
2	edith	paul	paul.edith@gmail.com

Saisir manuellement les inscriptions suivantes :

- L'étudiant **aude debece** aux enseignements de SLAM
- L'étudiant **paul edith** aux enseignements de SISR

Vérifier par une requête SQL l'inscription des étudiants aux différents enseignements (prénom et nom de l'étudiant, nom de la spécialité, intitulé de l'enseignement).

Le résultat doit correspondre aux 4 occurrences suivantes :

Sortie de données | Expliquer (Explain) | Messages | Historique

	prenom character va	nom character varying(25)	nom_specialite character varying(25)	intitule character varying(60)
1	aude	debece	SLAM	SGBD
2	aude	debece	SLAM	PROG
3	paul	edith	SISR	SERVICES
4	paul	edith	SISR	ADMIN

Query | Query History

```
1  set schema 'TX_shema';
2
3  INSERT INTO etudiant Values (1,'debece','aude','aude.debece@gmail.com');
4  INSERT INTO etudiant Values (2,'edith','paul','paul.edith@gmail.com');
5
6  SELECT * FROM etudiant;
```

Data Output | Messages | Notifications

SQL

	etudiant_id [PK] integer	nom character varying (25)	prenom character varying (25)	email character varying (100)
1	1	debece	aude	aude.debece@gmail.com
2	2	edith	paul	paul.edith@gmail.com

Insertion des données des étudiants

```
1  set schema 'TX_shema';
2
3  INSERT INTO inscription Values (
4      (SELECT etudiant_id FROM etudiant WHERE etudiant.nom='debece'),
5      (SELECT enseignement_id FROM enseignement WHERE enseignement.intitule='SGBD'),
6      (SELECT specialite_id FROM specialite WHERE nom_specialite='SLAM'));
```

Data Output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 81 msec.
```

Insertion de l'inscriptions pour debece à l'enseignement SGBD de la spécialité SLAM

Query Query History

```
1  set schema 'TX_shema';
2
3  INSERT INTO inscription Values (
4      (SELECT etudiant_id FROM etudiant WHERE etudiant.nom='debece'),
5      (SELECT enseignement_id FROM enseignement WHERE enseignement.intitule='PROG'),
6      (SELECT specialite_id FROM specialite WHERE nom_specialite='SLAM'));
7
8  INSERT INTO inscription Values (
9      (SELECT etudiant_id FROM etudiant WHERE etudiant.nom='edith'),
10     (SELECT enseignement_id FROM enseignement WHERE enseignement.intitule='SERVICES'
11     (SELECT specialite_id FROM specialite WHERE nom_specialite='SISR'));
12
13 INSERT INTO inscription Values (
14     (SELECT etudiant_id FROM etudiant WHERE etudiant.nom='edith'),
15     (SELECT enseignement_id FROM enseignement WHERE enseignement.intitule='ADMIN'),
16     (SELECT specialite_id FROM specialite WHERE nom_specialite='SISR'));
```

Insertion des données pour les autres inscriptions

Query Query History

```
1  set schema 'TX_shema';
2
3  SELECT etudiant.prenom, etudiant.nom, specialite.nom_specialite, enseignement.intitule
4  FROM inscription
5  natural join etudiant
6  natural join enseignement
7  natural join specialite
```

Data Output Messages Notifications

	prenom character varying (25)	nom character varying (25)	nom_specialite character varying (25)	intitule character varying (60)
1	aude	debece	SLAM	SGBD
2	aude	debece	SLAM	PROG
3	paul	edith	SISR	SERVICES
4	paul	edith	SISR	ADMIN

Résultat de la requête à la suite des insertions des données des inscriptions

A faire en bonus:

On voudrait donner la possibilité de gérer la **note** d'un étudiant à un enseignement en ajoutant un attribut **Note**, de type **entier** à la table **Inscription**
Tester en saisissant des notes et en calculant la moyenne générale par étudiant par exemple

Query Query History

```
1  set schema 'TX_shema';
2
3  ALTER TABLE inscription
4  ADD note FLOAT;
5  SELECT * from inscription
```

Data Output Messages Notifications

	etudiant_id [PK] integer	enseignement_id [PK] integer	specialite_id [PK] integer	note double precision
1	1	1	1	[null]
2	1	2	1	[null]
3	2	3	2	[null]
4	2	4	2	[null]

Insertion de la colonne note

Query Query History

```
1  set schema 'TX_shema';
2
3  SELECT etudiant.prenom, etudiant.nom, specialite.nom_specialite, enseignement.intitule, note
4  FROM inscription
5  natural join etudiant
6  natural join enseignement
7  natural join specialite
```

Data Output Messages Notifications

	prenom character varying (25)	nom character varying (25)	nom_specialite character varying (25)	intitule character varying (60)	note double precision
1	aude	debece	SLAM	SGBD	15.5
2	aude	debece	SLAM	PROG	12.25
3	paul	edith	SISR	SERVICES	14.75
4	paul	edith	SISR	ADMIN	13

Insertion des notes

Query | Query History

```
1  set schema 'TX_shema';
2
3  SELECT etudiant.prenom, etudiant.nom, specialite.nom_specialite, AVG(note) as "Moyenne"
4  FROM inscription
5  natural join etudiant
6  natural join specialite
7  group by etudiant.prenom, etudiant.nom, specialite.nom_specialite;
```

Data Output | Messages | Notifications

	prenom character varying (25)	nom character varying (25)	nom_specialite character varying (25)	Moyenne double precision
1	paul	edith	SISR	13.875
2	aude	debece	SLAM	13.875

Etape 6 – Compléter le schéma de la base de données

Pour compléter la base de données, on désire gérer l'emploi du temps des enseignants qui dispensent les différents enseignements.
On doit pouvoir notamment enregistrer les jours et heures des différents enseignements.

Un extrait des données à gérer est présenté dans le tableau suivant :

SLAM 3	COURSIER	LUNDI	10 h
SLAM 4	DUPRE	MARDI	8 h
SLAM 5	MARC	MARDI	11 h
SISR 4	JACOB	LUNDI	16 h
PPE SLAM	COURSIER	JEUDI	8 h
PPE SLAM	DUPRE	JEUDI	8 h
PPE SISR	FOULQUIER	JEUDI	8 h

A faire :
Effectuer les opérations nécessaires pour assurer la gestion de ces données.
Faire afficher les enseignements du Jeudi (nom enseignant, intitule enseignement, spécialité)

Query Query History

```
2  CREATE TABLE EMPLOIE
3     (
4      EMPLOIE_ID int4 NOT NULL  ,
5      JOUR DATE NOT NULL  ,
6      HEURE TIME NOT NULL  ,
7      ENSEIGNANT_ID int4 NOT NULL  ,
8      ENSEIGNEMENT_ID int4 NOT NULL,
9      SPECIALITE_ID int4 NOT NULL
10 ,   CONSTRAINT PK_EMPLOIE_ID PRIMARY KEY (EMPLOIE_ID)
11    ) ;
12
13 CREATE  INDEX I_FK_EMPLOIE_ENSEIGNANT
14      ON EMPLOIE (ENSEIGNANT_ID)
15     ;
16 CREATE  INDEX I_FK_EMPLOIE_ENSEIGNEMENT
17      ON EMPLOIE (ENSEIGNEMENT_ID)
18     ;
19 CREATE  INDEX I_FK_EMPLOIE_specialite
20      ON EMPLOIE (SPECIALITE_ID)
21     ;
22
23
24 -- ------------------------------------------------
25 --        CREATION DES REFERENCES DE TABLE
26 -- ------------------------------------------------
27
28
29 ALTER TABLE EMPLOIE ADD
30      CONSTRAINT FK_EMPLOIE_ENSEIGNANT
31           FOREIGN KEY (ENSEIGNANT_ID)
32                REFERENCES ENSEIGNANT (ENSEIGNANT_ID);
33
34 ALTER TABLE EMPLOIE ADD
35      CONSTRAINT FK_EMPLOIE_SPECIALITE
36           FOREIGN KEY (SPECIALITE_ID)
```

Data Output Messages Notifications

```
ALTER TABLE

Query returned successfully in 77 msec.
```

Création de la nouvelle table emploi

Query | Query History

```
1  set schema 'TX_shema';
2
3  INSERT INTO enseignant VALUES
4  ('4',
5  (SELECT formation_id FROM formation WHERE nom_formation='BTS SIO'),
6  'MARC',
7  'PHILLIPE'
8  );
9
10 INSERT INTO enseignant VALUES
11 ('5',
12 (SELECT formation_id FROM formation WHERE nom_formation='BTS SIO'),
13 'JACOB',
14 '???'
15 );
16
17 INSERT INTO specialite VALUES
18 ('3',
19 (SELECT formation_id FROM formation WHERE nom_formation='BTS SIO'),
20 (SELECT enseignant_id from enseignant WHERE nom='COURSIER'),
21 'SLAM 3');
22
23 INSERT INTO specialite VALUES
24 ('4',
25 (SELECT formation_id FROM formation WHERE nom_formation='BTS SIO'),
26 (SELECT enseignant_id from enseignant WHERE nom='DUPRE'),
27 'SLAM 4');
28
29 INSERT INTO specialite VALUES
30 ('5',
31 (SELECT formation_id FROM formation WHERE nom_formation='BTS SIO'),
32 (SELECT enseignant_id from enseignant WHERE nom='MARC'),
33 'SLAM 5');
34
35 INSERT INTO specialite VALUES
36 ('6'
```

Data Output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 78 msec.
```

Total rows: 2 | Query complete 00:00:00.078

Query Query History

```
1 SELECT * FROM "TX_shema".enseignant
2 ORDER BY enseignant_id ASC
```

Data Output Messages Notifications

	enseignant_id [PK] integer	formation_id integer	nom character varying (25)	prenom character varying (25)
1	1	1	COURSIER	STEFAN
2	2	1	DUPRE	PATRICK
3	3	1	FOULQUIER	LAURENCE
4	4	1	MARC	PHILLIPE
5	5	1	JACOB	???

Insertion des données des nouveaux enseignants

Query Query History

```
1 SELECT * FROM "TX_shema".specialite
2 ORDER BY specialite_id ASC
```

Data Output Messages Notifications

	specialite_id [PK] integer	formation_id integer	enseignant_id integer	nom_specialite character varying (25)
1	1	1	1	SLAM
2	2	1	3	SISR
3	3	1	1	SLAM 3
4	4	1	2	SLAM 4
5	5	1	4	SLAM 5
6	6	1	5	SISR 4
7	7	1	1	PPE SLAM
8	8	1	2	PPE SLAM
9	9	1	3	PPE SISR

Insertion des données des nouveaux spécialités selon l'enseignant

Query Query History

```
2  INSERT INTO emploie VALUES
3  ('1',
4  'LUNDI',
5  '10:00',
6  (SELECT enseignant_id from enseignant WHERE nom='COURSIER'),
7  (SELECT specialite_id from specialite WHERE nom_specialite='SLAM 3')),
8  ('2',
9  'MARDI',
10 '8:00',
11 (SELECT enseignant_id from enseignant WHERE nom='DUPRE'),
12 (SELECT specialite_id from specialite WHERE nom_specialite='SLAM 4')),
13 ('3',
14 'MARDI',
15 '11:00',
16 (SELECT enseignant_id from enseignant WHERE nom='MARC'),
17 (SELECT specialite_id from specialite WHERE nom_specialite='SLAM 5')),
18 ('4',
19 'LUNDI',
20 '16:00',
21 (SELECT enseignant_id from enseignant WHERE nom='JACOB'),
22 (SELECT specialite_id from specialite WHERE nom_specialite='SISR 4')),
23 ('5',
24 'JEUDI',
25 '8:00',
26 (SELECT enseignant_id from enseignant WHERE nom='COURSIER'),
27 7),
28 ('6',
29 'JEUDI',
30 '8:00',
31 (SELECT enseignant_id from enseignant WHERE nom='DUPRE'),
32 8),
33 ('7',
34 'JEUDI',
35 '8:00',
36 (SELECT enseignant_id from enseignant WHERE nom='FOULQUIER'),
37 (SELECT specialite_id from specialite WHERE nom_specialite='PPE SISR'));
```

Data Output Messages Notifications

```
INSERT 0 7

Query returned successfully in 81 msec.
```

Insertion des données dans la table emploie

Query Query History

```
1  set schema 'TX_shema';
2  select jour, heure, enseignant.nom, specialite.nom_specialite from emploie
3  natural join enseignant
4  natural join specialite
5  where jour='JEUDI'
```

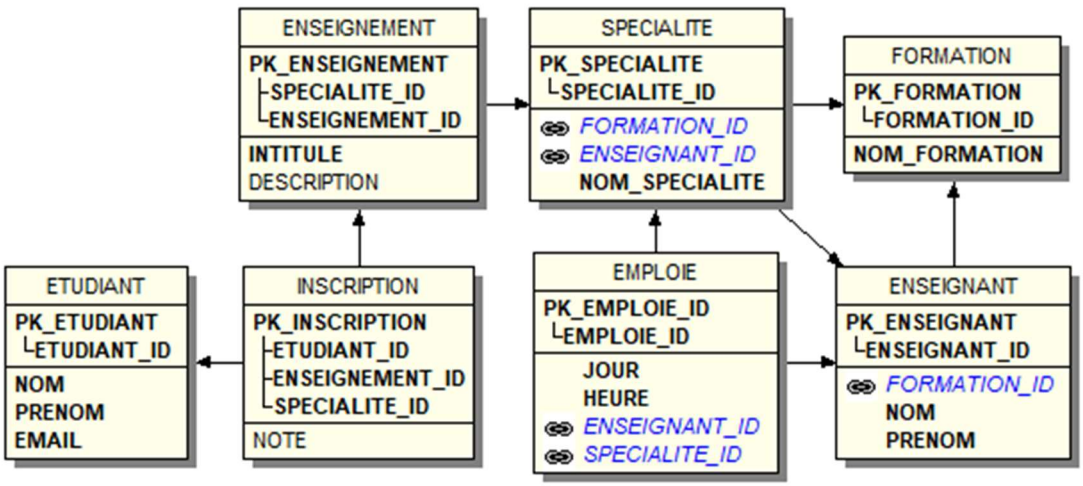
Data Output Messages Notifications

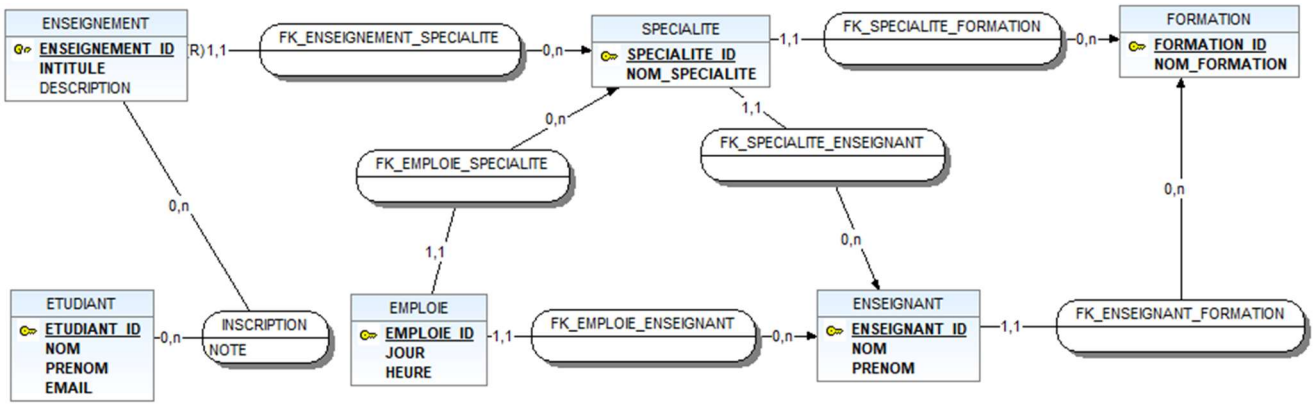
	jour character varying (20)	heure time without time zone	nom character varying (25)	nom_specialite character varying (25)
1	JEUDI	08:00:00	COURSIER	PPE SLAM
2	JEUDI	08:00:00	DUPRE	PPE SLAM
3	JEUDI	08:00:00	FOULQUIER	PPE SISR

Etape 7 – Bilan

On vous demande de produire un schéma des données résultant de la mise en place de la base de données au cours des étapes précédentes (Etape 3 à Etape 6).

A faire :
Avec l'outil de sauvegarde de la base de données sous PgAdmin, récupérer le fichier qui permettra de réaliser un **Reverse Engineering** de la base de données avec WinDesign pour visualiser le schéma de données sous la forme entité association.




ENSEIGNEMENT
ENSEIGNEMENT_ID
INTITULE
DESCRIPTION
(R) 1,1
FK_ENSEIGNEMENT_SPECIALITE
0,n
SPECIALITE
SPECIALITE_ID
NOM_SPECIALITE
1,1
FK_SPECIALITE_FORMATION
0,n
FORMATION
FORMATION_ID
NOM_FORMATION
0,n
FK_EMPLOIE_SPECIALITE
1,1
FK_SPECIALITE_ENSEIGNANT
0,n
0,n
1,1
0,n
ETUDIANT
ETUDIANT_ID
NOM
PRENOM
EMAIL
0,n
INSCRIPTION
NOTE
EMPLOIE
EMPLOIE_ID
JOUR
HEURE
1,1
FK_EMPLOIE_ENSEIGNANT
0,n
ENSEIGNANT
ENSEIGNANT_ID
NOM
PRENOM
1,1
FK_ENSEIGNANT_FORMATION