

**Module SLAM 3      TP3**

Propriétés	Description
<b>Intitulé</b>	<b>INIT_POSTGRESQL</b>
<b>Outils</b>	<ul style="list-style-type: none"> <li>• A.G.L WIN'DESIGN</li> <li>• SGBD POSTGRESQL</li> </ul>
<b>Durée estimée en heures</b>	4 Heures
<b>Savoir-faire module SLAM3</b>	<ul style="list-style-type: none"> <li>• Concevoir une base de données</li> <li>• Valider un schéma de base de données</li> <li>• Programmer dans l'environnement de développement associé à un SGBD</li> </ul>
<b>Savoirs Module SLAM3</b>	<ul style="list-style-type: none"> <li>• Modèles de représentation des données</li> <li>• Langage de programmation associé à un SGBD</li> </ul>
<b>Documents joints</b>	Fiche d'exploitation pédagogique Annexes
<b>Réception</b>	Mise en place et exploitation du SGBD Postgresql
<b>Equipe</b>	Seul    ×    Par équipe de ... <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 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'installateurs pour Windows. Les plus usités sont créés par des entreprises ( BigSql, EnterpriseDB, ... ) d'autres sont créés par la communauté.

L'installateur "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.

### **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 tâches

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	<b>help</b> ou <b>\ ?</b>
Afficher la liste des bases de données	<b>\l</b>
Afficher la liste des tables	<b>\dt</b>
Afficher la liste des utilisateurs	<b>\du</b>
Afficher la liste des langages installés	<b>\dL</b>
Quitter <b>psql</b>	


## • 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)

>  pgAdmin 4 (4)

0,3%

218,5 Mo

0,1 Mo/s

0 Mbits/s

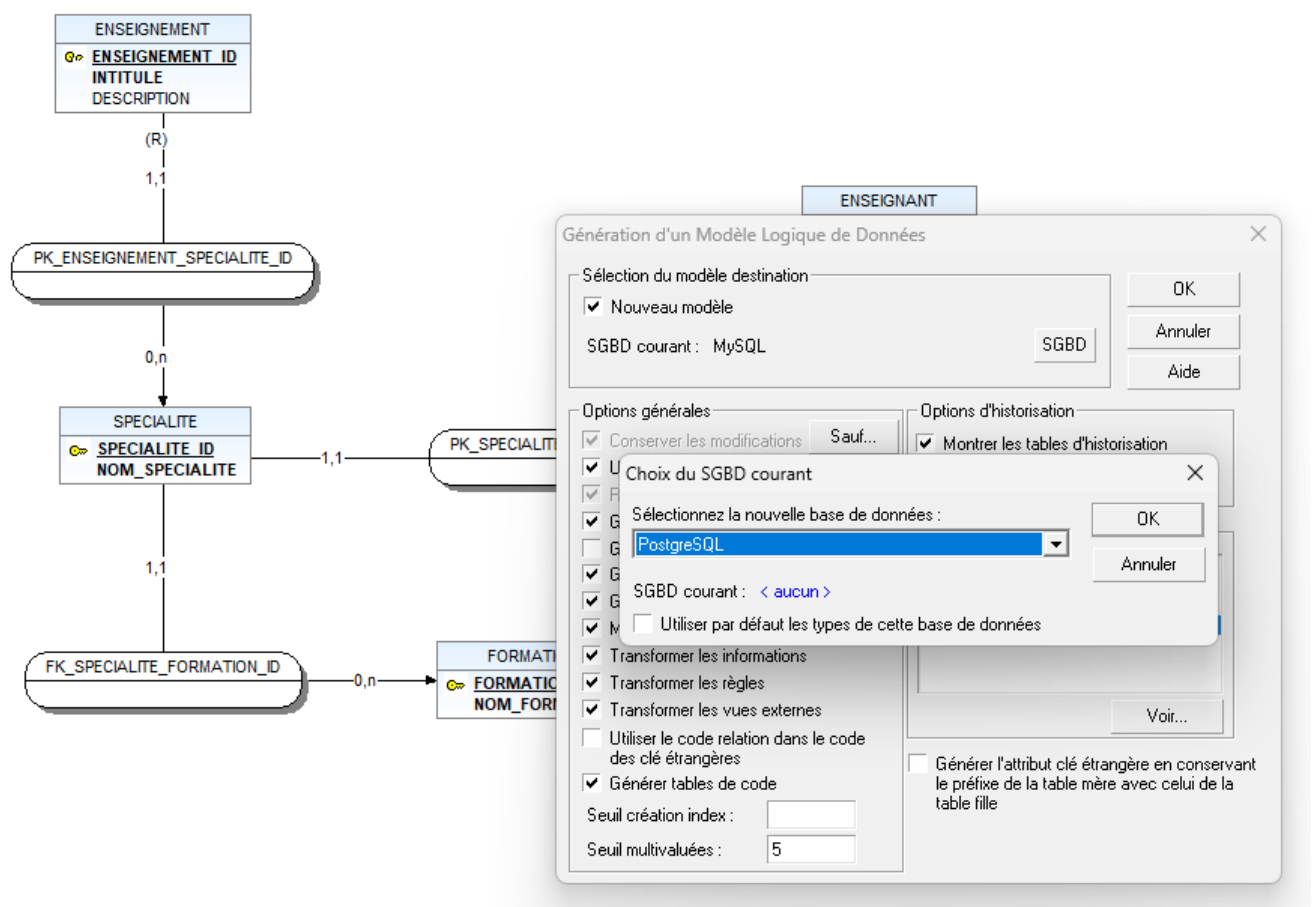
Etape 3 - Créer une base de données PostgreSQL

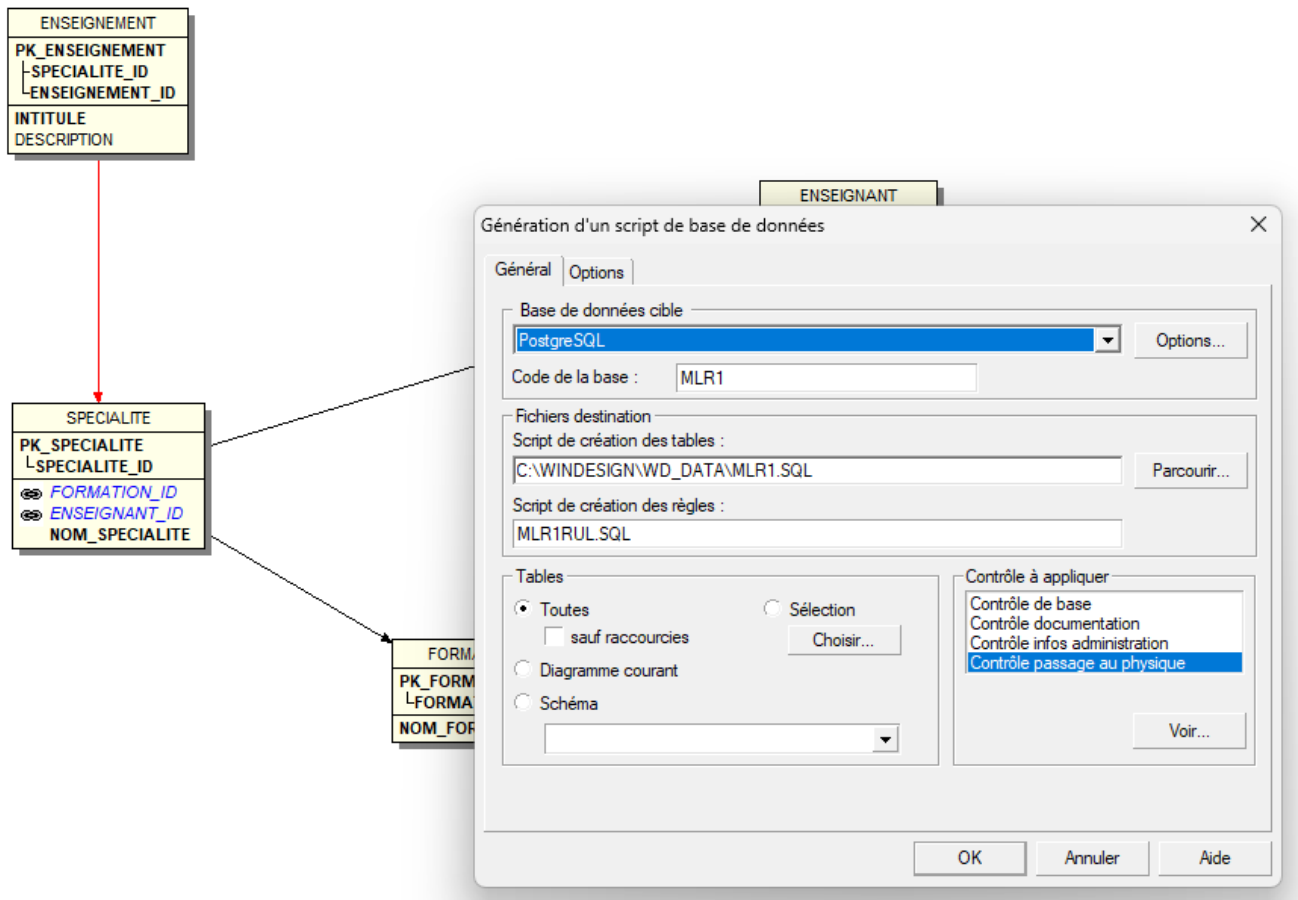
La base de données concerne une partie de la structure des enseignements en BTSSIO.

Le schéma des données a été réalisé avec WINDESIGN.

### A faire :

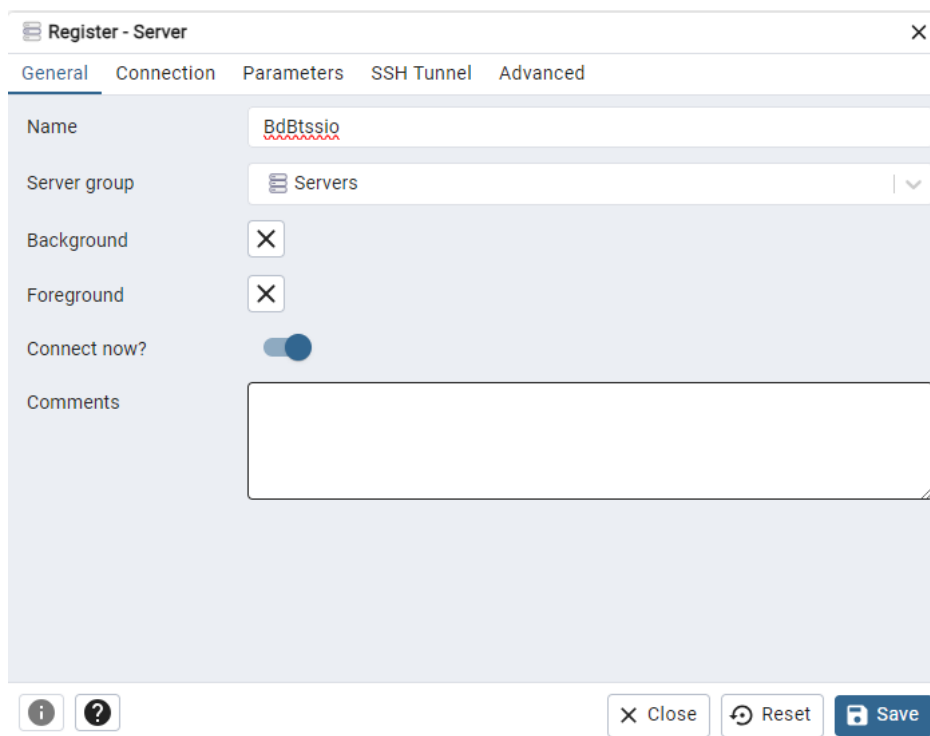
- Récupérer le fichier .mcd correspondant appelé **BdBtssioMCD**
- Avec **WINDESIGN** : générer le script pour *Postgresql*





Avec **PgAdmin** :

créer la base de données **BdBtssio**  
exécuter le script dans l'éditeur SQL



Register - Server

×

General Connection Parameters SSH Tunnel Advanced

Host name/address

172.28.38.150

Port

5432

Maintenance database

postgres

Username

postgres

Kerberos authentication?

☐

Password

.....

Save password?

☒

Role

Service

i ?

Close

Reset

Save

Create - Database


×

General Definition Security Parameters Advanced SQL

Database

MPP-BdBtssio

Owner

 postgres

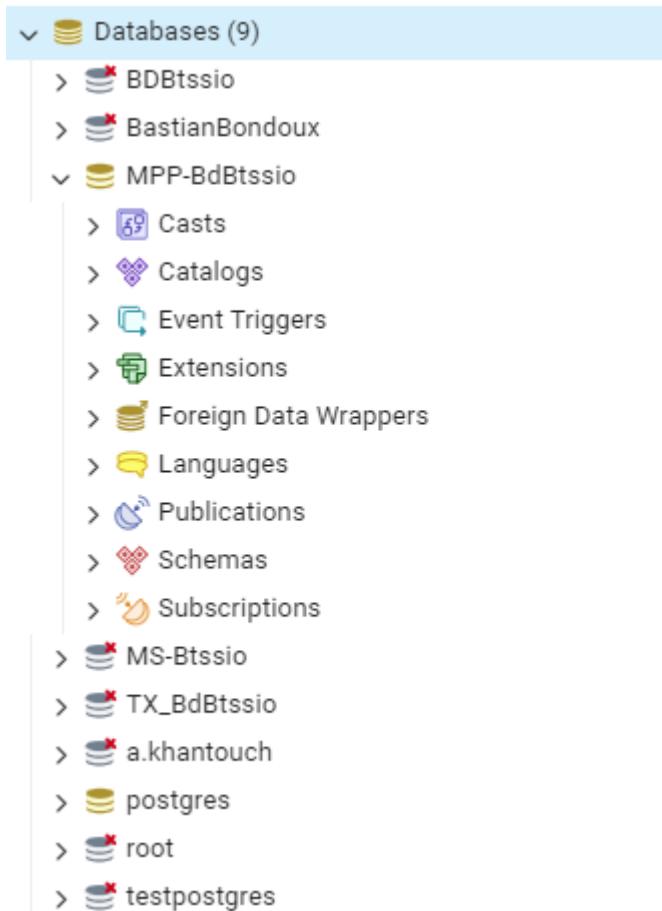
Comment

i ?

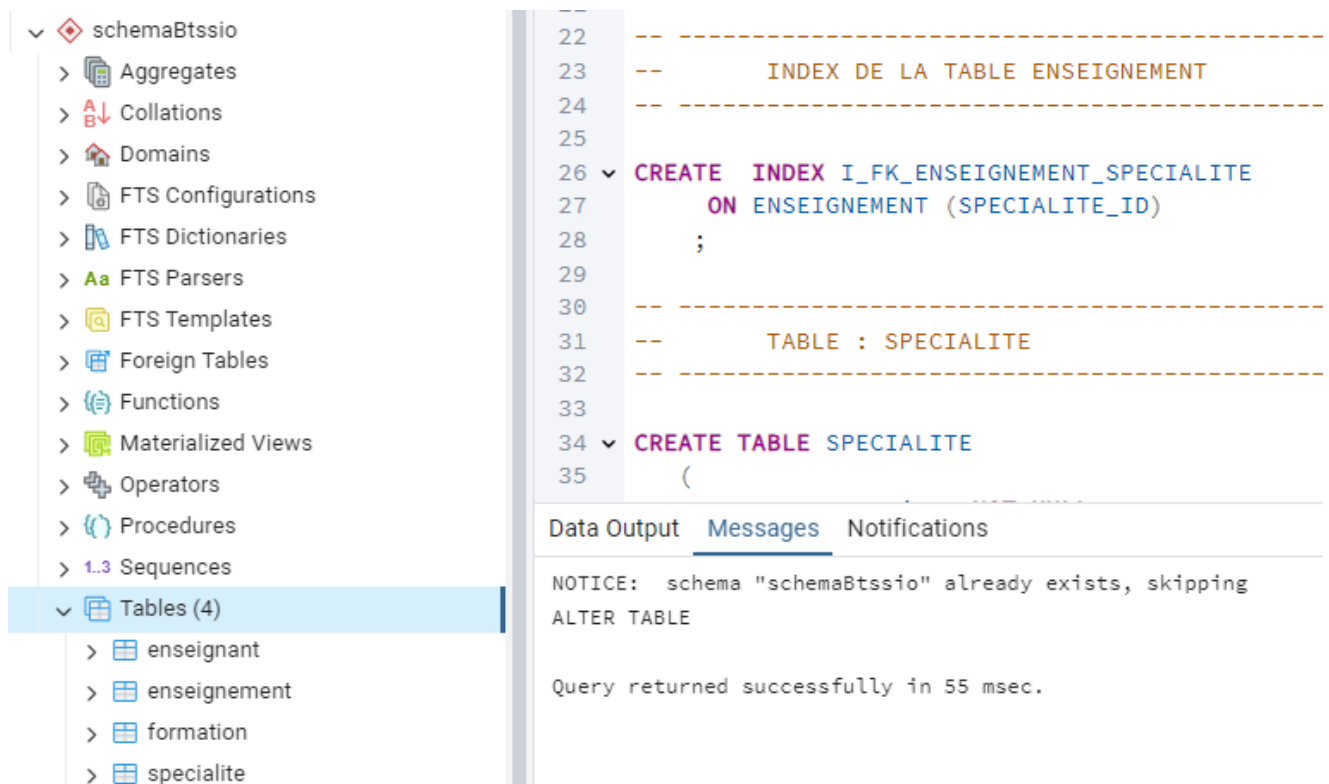
Close

Reset

Save



- Vérifier la création effective des tables dans le schéma de 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.

Data Output Messages Notifications

```
ERROR: Key (specialite_id)=(3) is not present in table "specialite".insert or update on table "enseignement" violates foreign key constraint "fk_enseignement_specialite"

ERROR: insert or update on table "enseignement" violates foreign key constraint "fk_enseignement_specialite"
SQL state: 23503
Detail: Key (specialite_id)=(3) is not present in table "specialite".
```

L'insertion est impossible car il n'existe pas de specialite\_id qui est égal à 3.

Pour régler le problème, il faut inverser les '3' et les '4' par la sous requête.

```
16 INSERT INTO Enseignement VALUES ((SELECT Specialite_ID FROM Specialite WHERE Nom_Specialite='SLAM'),'3','SGBE
17 INSERT INTO Enseignement VALUES ((SELECT Specialite_ID FROM Specialite WHERE Nom_Specialite='SLAM'),'4','PROG
18 INSERT INTO Enseignement VALUES ((SELECT Specialite_ID FROM Specialite WHERE Nom_Specialite='SISR'),'3','SERV
19 INSERT INTO Enseignement VALUES ((SELECT Specialite_ID FROM Specialite WHERE Nom_Specialite='SISR'),'4','ADMJ
20
21
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 188 msec.

### A faire :

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

- Liste des enseignants (nom et prénom)

Query

Query History

1











2



SELECT nom, prenom FROM "schemaBtssio".enseignant

Data Output

Messages

Notifications



	nom character varying (25) 	prenom character varying (25) 
1	COURSIER	STEFAN
2	DUPRE	PATRICK
3	FOULQUIER	LAURENCE

Requête : SELECT nom, prenom FROM "schemaBtssio".enseignant ;

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

```

2  SELECT intitule, description, specialite.nom_specialite FROM enseignement
3  NATURAL JOIN specialite
4  WHERE specialite.nom_specialite = 'SLAM'

```

Data Output Messages Notifications

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

Requête : SELECT intitule, description, specialite.nom\_specialite FROM enseignement  
 NATURAL JOIN specialite  
 WHERE specialite.nom\_specialite = 'SLAM'



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

```
2 SELECT specialite.nom_specialite, enseignant.nom, enseignant.prenom FROM specialite
3 NATURAL JOIN enseignant
4
```

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

Requête :

```
SELECT specialite.nom_specialite, enseignant.nom, enseignant.prenom FROM
specialite
NATURAL JOIN enseignant
```

- Nombre d'enseignements par spécialité (nom\_spécialité, nombre)

```
3 SELECT specialite.nom_specialite, count(enseignant.enseignant_id) FROM specialite
4 NATURAL JOIN enseignant
5 GROUP BY specialite.nom_specialite
```

Requête : SELECT  
specialite.nom\_specialite,  
count(enseignant.enseignant\_id)  
FROM specialite  
NATURAL JOIN enseignant  
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

```

3 CREATE TABLE Etudiant
4 (
5     Etudiant_ID integer NOT NULL,
6     Nom varchar(25) NOT NULL,
7     Prenom Varchar(25) NOT NULL,
8     Email Varchar(100) DEFAULT NULL,
9     CONSTRAINT PK_Etudiant PRIMARY KEY (Etudiant_ID) -- Définition de la clé primaire
10 );
11
12 CREATE TABLE Inscription
13 (
14     Etudiant_ID integer NOT NULL,
15     Enseignement_ID integer NOT NULL,
16     Specialite_ID integer NOT NULL,
17     CONSTRAINT PK_Inscription PRIMARY KEY (Etudiant_ID,Enseignement_ID,Specialite_ID), -- Définition de la clé primaire
18     CONSTRAINT "FK_Inscription_Etudiant" FOREIGN KEY (Etudiant_ID) REFERENCES Etudiant (Etudiant_ID) ON UPDATE RESTRICT ON DEL
19     CONSTRAINT "FK_Inscription_Enseignement" FOREIGN KEY (Enseignement_ID,Specialite_ID) REFERENCES Enseignement (Enseignement
20     CONSTRAINT "UN_Inscription" UNIQUE (Etudiant_ID,Enseignement_ID,Specialite_ID) -- pour vérifier que le triplet est unique
21 );
22

```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 83 msec.

Insérer manuellement les occurrences suivantes d'étudiants avec PgAdmin , ou avec Psql

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

```

1 SET SCHEMA 'schemaBtssio';
2 SELECT * FROM etudiant

```

Requête 1 : INSERT INTO etudiant  
VALUES(1, 'debece', 'aude',  
'aude.debece@gmail.com');

Requête 2 : INSERT INTO etudiant  
VALUES(2, 'edith', 'paul',  
'paul.edith@gmail.com');

Data Output Messages Notifications

	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

Saisir manuellement les inscriptions suivantes :

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

Query Query History

```

1 SET SCHEMA 'schemaBtssio';
2
3 INSERT INTO inscription VALUES((SELECT etudiant_id FROM etudiant WHERE nom = 'debec'),
4 (SELECT enseignement_id FROM enseignement WHERE intitule = 'SGBD'),
5 (SELECT specialite_id FROM specialite WHERE nom_specialite = 'SLAM'));
6
7 INSERT INTO inscription VALUES((SELECT etudiant_id FROM etudiant WHERE nom = 'debec'),
8 (SELECT enseignement_id FROM enseignement WHERE intitule = 'PROG'),
9 (SELECT specialite_id FROM specialite WHERE nom_specialite = 'SLAM'));
10
11 INSERT INTO inscription VALUES((SELECT etudiant_id FROM etudiant WHERE nom = 'edith'),
12 (SELECT enseignement_id FROM enseignement WHERE intitule = 'SERVICES'),
13 (SELECT specialite_id FROM specialite WHERE nom_specialite = 'SISR'));
14
15 INSERT INTO inscription VALUES((SELECT etudiant_id FROM etudiant WHERE nom = 'edith'),
16 (SELECT enseignement_id FROM enseignement WHERE intitule = 'ADMIN'),
17 (SELECT specialite_id FROM specialite WHERE nom_specialite = 'SISR'));
18
19 SELECT * FROM inscription;

```

Data Output Messages Notifications

	etudiant_id [PK] integer	enseignement_id [PK] integer	specialite_id [PK] integer
1	1	3	1
2	1	4	1
3	2	3	2
4	2	4	2

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

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

```

3 SELECT etudiant.prenom, etudiant.nom, specialite.nom_specialite, enseignement.intitule AS intitule FROM inscription
4 NATURAL JOIN etudiant
5 NATURAL JOIN specialite
6 NATURAL JOIN enseignement;
7

```

Data Output Messages Notifications

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

Requête : SELECT etudiant.prenom, etudiant.nom, specialite.nom\_specialite, enseignement.intitule  
 AS intitule FROM inscription  
 NATURAL JOIN etudiant  
 NATURAL JOIN specialite  
 NATURAL JOIN enseignement;

### **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

## **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.  
 Création de la table de l'emploi du temps :

```

3 CREATE TABLE emploiutemps(
4     emploi_id Integer NOT NULL,
5     enseignant_id Integer NOT NULL,
6     enseignement_id Integer NOT NULL,
7     specialite_id Integer NOT NULL,
8     jour VARCHAR(8) NOT NULL,
9     horaire time NOT NULL,
10    CONSTRAINT PK_Emploi PRIMARY KEY (emploi_id),
11    CONSTRAINT "PK_Emploi_Enseignant" FOREIGN KEY (enseignant_id) REFERENCES enseignant (enseignant_id),
12    CONSTRAINT "PK_Emploi_Enseignement" FOREIGN KEY (enseignement_id,specialite_id) REFERENCES enseignement (enseignement_id
13    CONSTRAINT "PK_Emploi_Specialite" FOREIGN KEY (specialite_id) REFERENCES specialite (specialite_id)
14 );
  
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 70 msec.

Requête :

```
CREATE TABLE emploiutemps(
    emploi_id Integer NOT NULL,
    enseignant_id Integer NOT NULL,
    enseignement_id Integer NOT NULL,
    specialite_id Integer NOT NULL,
    jour VARCHAR(8) NOT NULL,
    horaire time NOT NULL,
    CONSTRAINT PK_Emploi PRIMARY KEY (emploi_id),
    CONSTRAINT "PK_Emploi_Enseignant" FOREIGN KEY (enseignant_id) REFERENCES
enseignant (enseignant_id),
    CONSTRAINT "PK_Emploi_Enseignement" FOREIGN KEY (enseignement_id,
specialite_id) REFERENCES enseignement (enseignement_id, specialite_id),
    CONSTRAINT "PK_Emploi_Specialite" FOREIGN KEY (specialite_id) REFERENCES
specialite (specialite_id)
);
```

Insertion des données :

```
INSERT INTO enseignant VALUES(4, (SELECT formation_id FROM formation WHERE
nom_formation = 'BTS SIO'), 'MARC', 'PHILLIPE');
INSERT INTO enseignant VALUES(5, (SELECT formation_id FROM formation WHERE
nom_formation = 'BTS SIO'), 'JACOB', 'TACOS');
```

```
INSERT INTO specialite VALUES(3, (SELECT formation_id FROM formation WHERE
nom_formation = 'BTS SIO'), (SELECT enseignant_id FROM enseignant WHERE nom ==
'COURSIER'), 'SLAM 3');
```

```
INSERT INTO specialite VALUES(4, (SELECT formation_id FROM formation WHERE
nom_formation = 'BTS SIO'), (SELECT enseignant_id FROM enseignant WHERE nom ==
'DUPRE'), 'SLAM 4');
```

```
INSERT INTO specialite VALUES(5, (SELECT formation_id FROM formation WHERE
nom_formation = 'BTS SIO'), (SELECT enseignant_id FROM enseignant WHERE nom ==
'MARC'), 'SLAM 5');
```

```
INSERT INTO specialite VALUES(6, (SELECT formation_id FROM formation WHERE
nom_formation = 'BTS SIO'), (SELECT enseignant_id FROM enseignant WHERE nom ==
'JACOB'), 'SISR 4');
```

```
INSERT INTO specialite VALUES(7, (SELECT formation_id FROM formation WHERE
nom_formation = 'BTS SIO'), (SELECT enseignant_id FROM enseignant WHERE nom ==
'COURSIER'), 'PPE SLAM');
```

```
INSERT INTO specialite VALUES(8, (SELECT formation_id FROM formation WHERE
nom_formation = 'BTS SIO'), (SELECT enseignant_id FROM enseignant WHERE nom ==
'DUPRE'), 'PPE SLAM');
```

```
INSERT INTO specialite VALUES(9, (SELECT formation_id FROM formation WHERE
nom_formation = 'BTS SIO'), (SELECT enseignant_id FROM enseignant WHERE nom ==
'FOULQUIER'), 'PPE SISR');
```

```

3 INSERT INTO enseignant VALUES(4, (SELECT formation_id FROM formation WHERE nom_formation = 'BTS SIO'), 'MARC', 'PHILLIPE');
4 INSERT INTO enseignant VALUES(5, (SELECT formation_id FROM formation WHERE nom_formation = 'BTS SIO'), 'JACOB', 'TACOS');
5
6 INSERT INTO specialite VALUES(3, (SELECT formation_id FROM formation WHERE nom_formation = 'BTS SIO'), (SELECT enseignant_id
7 INSERT INTO specialite VALUES(4, (SELECT formation_id FROM formation WHERE nom_formation = 'BTS SIO'), (SELECT enseignant_id
8 INSERT INTO specialite VALUES(5, (SELECT formation_id FROM formation WHERE nom_formation = 'BTS SIO'), (SELECT enseignant_id
9 INSERT INTO specialite VALUES(6, (SELECT formation_id FROM formation WHERE nom_formation = 'BTS SIO'), (SELECT enseignant_id
10 INSERT INTO specialite VALUES(7, (SELECT formation_id FROM formation WHERE nom_formation = 'BTS SIO'), (SELECT enseignant_id
11 INSERT INTO specialite VALUES(8, (SELECT formation_id FROM formation WHERE nom_formation = 'BTS SIO'), (SELECT enseignant_id
12 INSERT INTO specialite VALUES(9, (SELECT formation_id FROM formation WHERE nom_formation = 'BTS SIO'), (SELECT enseignant_id

```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 61 msec.

Insert dans la table emploiutemps :

```

1 SET SCHEMA 'schemaBtssio';
2 INSERT INTO emploiutemps VALUES(1, (SELECT enseignant_id FROM enseignant WHERE nom = 'COURSIER'), (SELECT specialite_id FROM
3 INSERT INTO emploiutemps VALUES(2, (SELECT enseignant_id FROM enseignant WHERE nom = 'DUPRE'), (SELECT specialite_id FROM spe
4 INSERT INTO emploiutemps VALUES(3, (SELECT enseignant_id FROM enseignant WHERE nom = 'MARC'), (SELECT specialite_id FROM spe
5 INSERT INTO emploiutemps VALUES(4, (SELECT enseignant_id FROM enseignant WHERE nom = 'JACOB'), (SELECT specialite_id FROM spe
6 INSERT INTO emploiutemps VALUES(5, (SELECT enseignant_id FROM enseignant WHERE nom = 'COURSIER'), (SELECT specialite_id FROM
7 INSERT INTO emploiutemps VALUES(6, (SELECT enseignant_id FROM enseignant WHERE nom = 'DUPRE'), (SELECT specialite_id FROM spe
8 INSERT INTO emploiutemps VALUES(7, (SELECT enseignant_id FROM enseignant WHERE nom = 'FOULQUIER'), (SELECT specialite_id FROM

```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 83 msec.

INSERT INTO emploiutemps VALUES(1, (SELECT enseignant\_id FROM enseignant WHERE nom = 'COURSIER'), (SELECT specialite\_id FROM specialite WHERE nom\_specialite = 'SLAM 3'), 'LUNDI', '10:00:00');

INSERT INTO emploiutemps VALUES(2, (SELECT enseignant\_id FROM enseignant WHERE nom = 'DUPRE'), (SELECT specialite\_id FROM specialite WHERE nom\_specialite = 'SLAM 4'), 'MARDI', '08:00:00');

INSERT INTO emploiutemps VALUES(3, (SELECT enseignant\_id FROM enseignant WHERE nom = 'MARC'), (SELECT specialite\_id FROM specialite WHERE nom\_specialite = 'SLAM 5'), 'MARDI', '11:00:00');

INSERT INTO emploiutemps VALUES(4, (SELECT enseignant\_id FROM enseignant WHERE nom = 'JACOB'), (SELECT specialite\_id FROM specialite WHERE nom\_specialite = 'SISR 4'), 'LUNDI', '16:00:00');

```
INSERT INTO emploidutemps VALUES(5, (SELECT enseignant_id FROM enseignant WHERE
nom = 'COURSIER'), (SELECT specialite_id FROM specialite WHERE nom_specialite = 'PPE
SLAM' LIMIT 1), 'JEUDI', '08:00:00');
```

```
INSERT INTO emploidutemps VALUES(6, (SELECT enseignant_id FROM enseignant WHERE
nom = 'DUPRE'), (SELECT specialite_id FROM specialite WHERE nom_specialite = 'PPE SLAM'
LIMIT 1), 'JEUDI', '08:00:00');
```

```
INSERT INTO emploidutemps VALUES(7, (SELECT enseignant_id FROM enseignant WHERE
nom = 'FOULQUIER'), (SELECT specialite_id FROM specialite WHERE nom_specialite = 'PPE
SISR'), 'JEUDI', '08:00:00');
```

Faire afficher les enseignements du Jeudi (nom enseignant, intitulé enseignement, spécialité)

Requête :

```
SELECT enseignant.nom, specialite.nom_specialite, emploidutemps.jour FROM emploidutemps
NATURAL JOIN enseignant
NATURAL JOIN specialite
WHERE emploidutemps.jour = 'JEUDI'
```

```
2  SELECT enseignant.nom, specialite.nom_specialite, emploidutemps.jour FROM emploidutemps
3  NATURAL JOIN enseignant
4  NATURAL JOIN specialite
5  WHERE emploidutemps.jour = 'JEUDI'
6
```

Data Output Messages Notifications

	nom character varying (25)	nom_specialite character varying (25)	jour character varying (8)
1	COURSIER	PPE SLAM	JEUDI
2	FOULQUIER	PPE SISR	JEUDI

## 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.

