

TD Migration de donnée simple

Niveau 1 :	2
Étape 1 : Installation des outils	2
Étape 3 : Migration avec DBeaver	7
Étape 4 : Vérification	13
 Niveau 2 :	 15
Étape 1 : Configuration initiale	15
Étape 2 : Préparation de la base de données MySQL	15
Étape 3 : Configuration de Flyway pour la migration	16
Étape 4 : Lancement de la migration avec Flyway	17
Étape 5 : Vérification de la migration	18
 Niveau 3	 19
Étape 1 : Créer des scripts de test	19
Étape 2 : Exécuter Flyway	20

Niveau 1 :

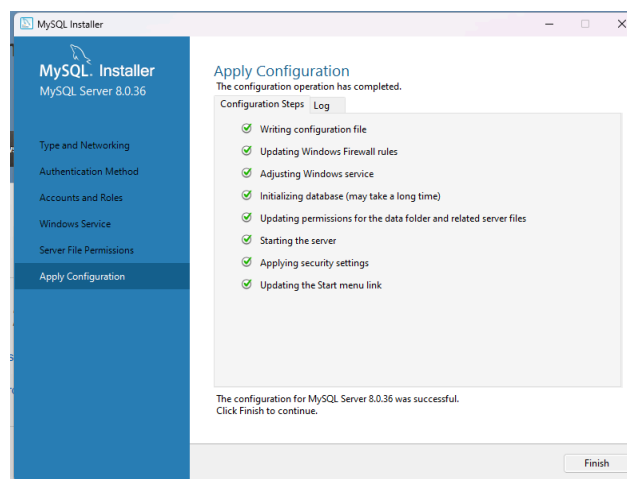
Étape 1 : Installation des outils

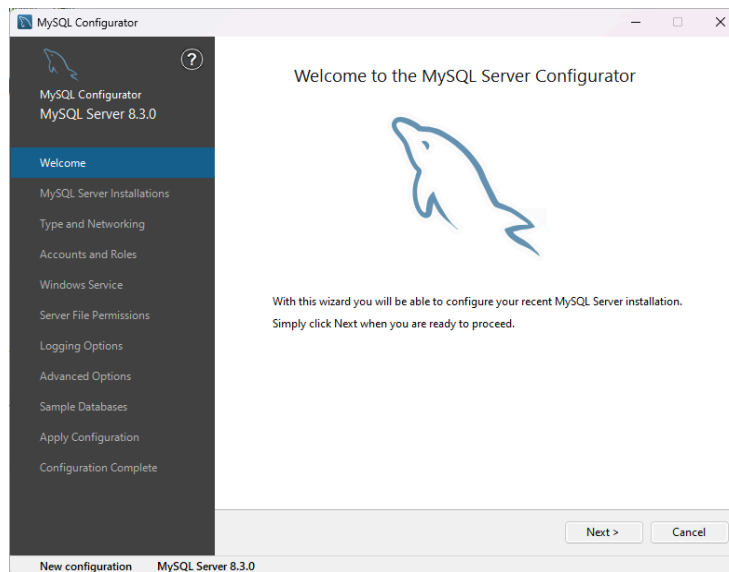
Pour l'installation des outils nous avons décidé de faire cela sur windows, la répartition des tâches étant que chaque personne devra s'occuper d'un niveau en ce qui concerne la rédaction, et en même temps faire l'ensemble des niveaux afin d'avoir un maximum de compréhension du sujet.

Les outils principaux étant MySQL et PostgreSQL, DBeaver qui s'occupera de la migration et Faker pour la génération de donnée factice.

1. Installation de Mysql

Nous nous sommes rendus sur le site web MySQL pour procéder à l'installation. Mysql est un système de gestion de base de données relationnelle open source, largement utilisé pour sa robustesse et sa fiabilité.



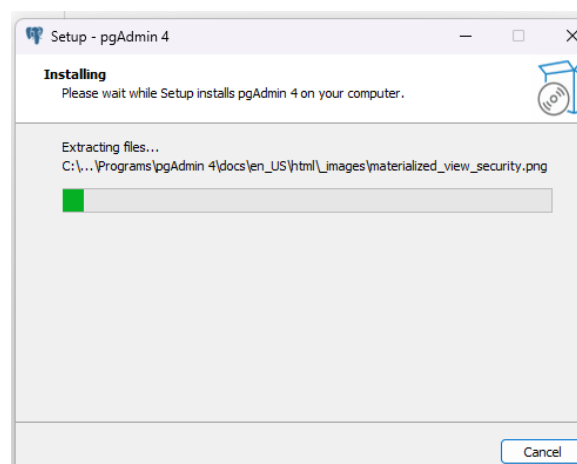


2. Installation de PostgreSQL et PgAdmin

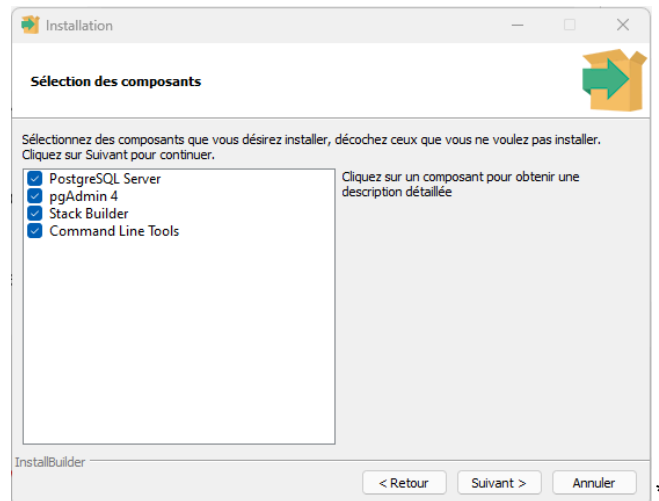
Pour l'installation nous nous sommes rendus sur le site web.

PostgreSQL est un système de gestion de base de données relationnelle open source offrant une grande extensibilité, des fonctionnalités avancées et une forte conformité aux normes SQL.

PgAdmin quant à elle est une interface graphique open source pour la gestion de bases de données PostgreSQL, offrant des fonctionnalités avancées telles que la visualisation des schémas, la création d'objets et la gestion des données.

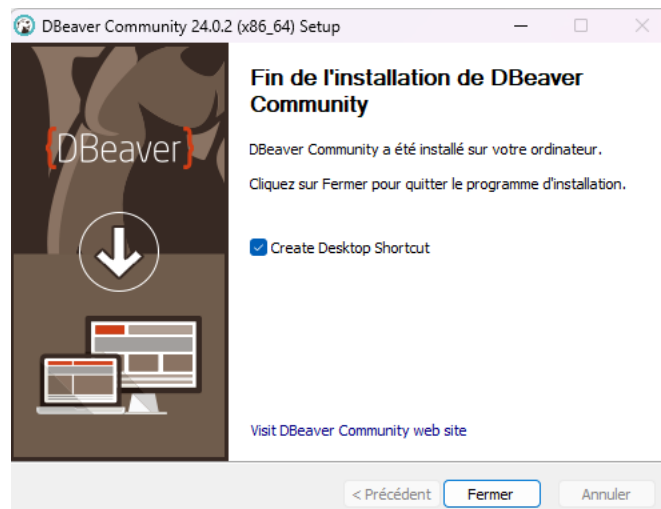


On peut voir que via l'installation de PostgreSQL, pgAdmin et d'autre composant ce sont installer.



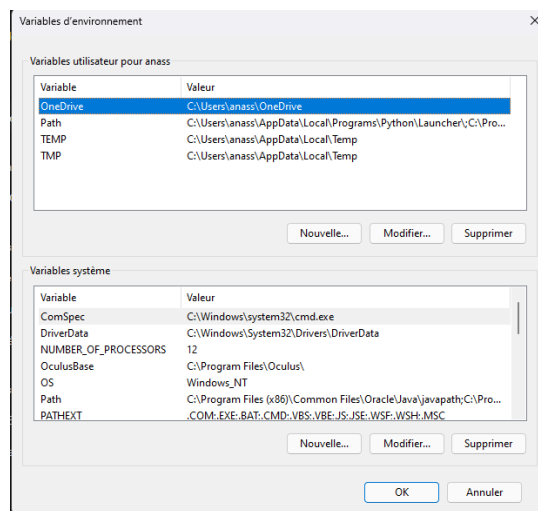
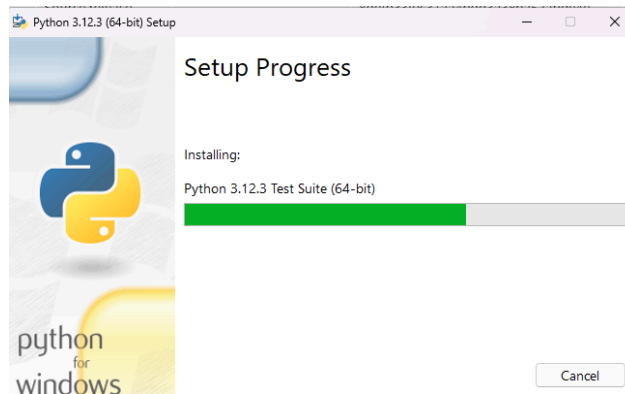
3. Installation et configuration de DBeaver

Pour DBeaver nous avons fait l'installation via leur site web. DBeaver est un client SQL universel open source prenant en charge de nombreux SGBD, offrant une interface intuitive, des fonctionnalités avancées de requêtage et de gestion de bases de données.



4. Installation et configuration de Faker

Pour l'installation de Faker il faut installer python car c'est une bibliothèque open source en python pour la génération de données factices, utile pour peupler les bases de données de test avec des données réalistes.



Une fois l'installation de python vérifiée, on peut donc dès à présent installer la librairie Faker avec cette commande.

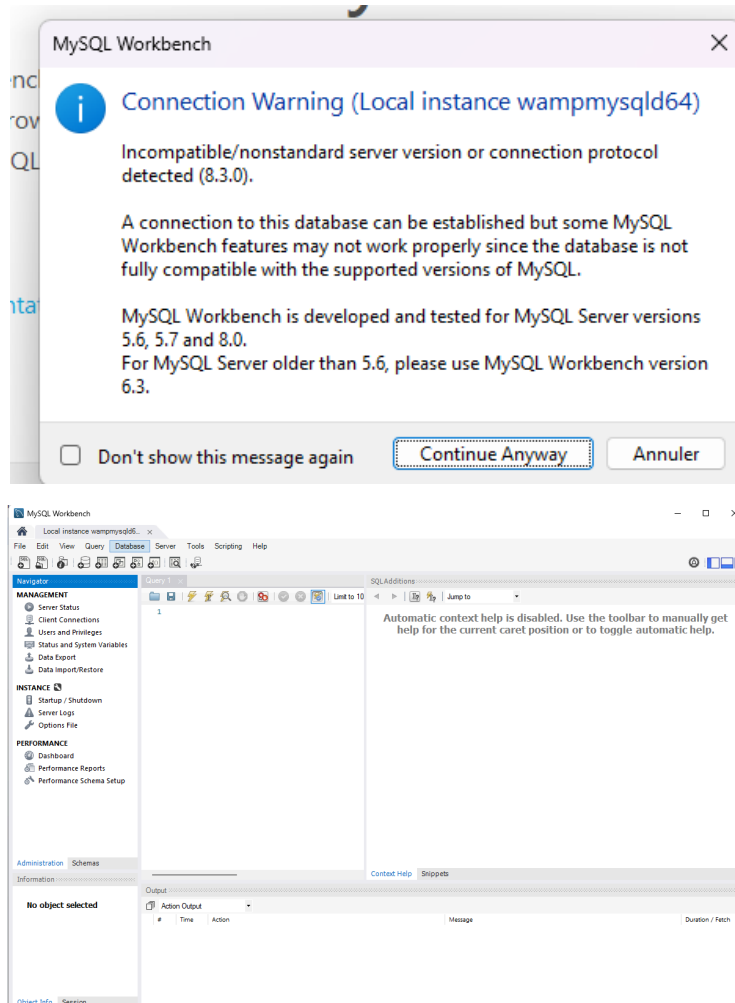
“pip install Faker”

```
C:\Users\anass>python
C:\Users\anass>pip install Faker
Defaulting to user installation because normal site-packages is not writeable
Collecting Faker
  Downloading Faker-24.11.0-py3-none-any.whl.metadata (15 kB)
Collecting python-dateutil>=2.4 (from Faker)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.4->Faker)
  Downloading six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Downloaded Faker-24.11.0-py3-none-any.whl (1.8 MB)
  1.8/1.8 MB 5.1 MB/s eta 0:00:00
Downloaded python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
  229.9/229.9 kB 7.1 MB/s eta 0:00:00
Downloaded six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, python-dateutil, Faker
WARNING: The script faker.exe is installed in 'C:\Users\anass\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\Local-packages\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed Faker-24.11.0 python-dateutil-2.9.0.post0 six-1.16.0
C:\Users\anass>
```

Étape 2 : Préparation de la base de données source

1. Création de la base de donnée MySQL

Nous allons donc commencer à créer la BD MySQL, pour cela nous avons le choix entre le workbench MySQL ou le CLI.



Nous avons observé que le workbench nous pose pas mal problème et pour ne pas perdre de temps nous avons décidé d'utiliser le CLI.

Nous avons créé la base donnée qui nous servira donc pour le côté MySQL avec la création de table ...

```
mysql> USE testdb
Database changed
mysql> CREATE TABLE Utilisateurs (
-> id INT AUTO_INCREMENT PRIMARY KEY,
-> Nom VARCHAR(255),
-> Prenom VARCHAR(255),
-> Email VARCHAR(255),
-> MotDePasse VARCHAR(255)
-> );
Query OK, 0 rows affected (0.01 sec)
mysql> |
```

2. Peuplement de la base de donnée MySQL

Pour le peuplement de la base de données on a donc fait cela avec Faker.

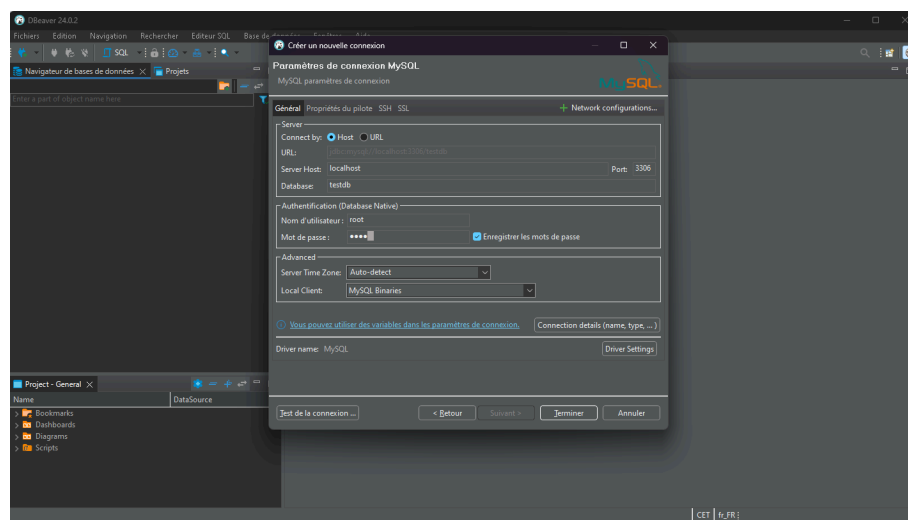
Avec les table que nous avons mis en place sur la BD MySQL, on a donc exécuté le script pour mettre en place des données factice dans celle-ci.

```
C:\Users\anass\Desktop\RIDA MSPR>python script_faker.py
Les données ont été insérées avec succès dans la base de données MySQL.
C:\Users\anass\Desktop\RIDA MSPR>|
```

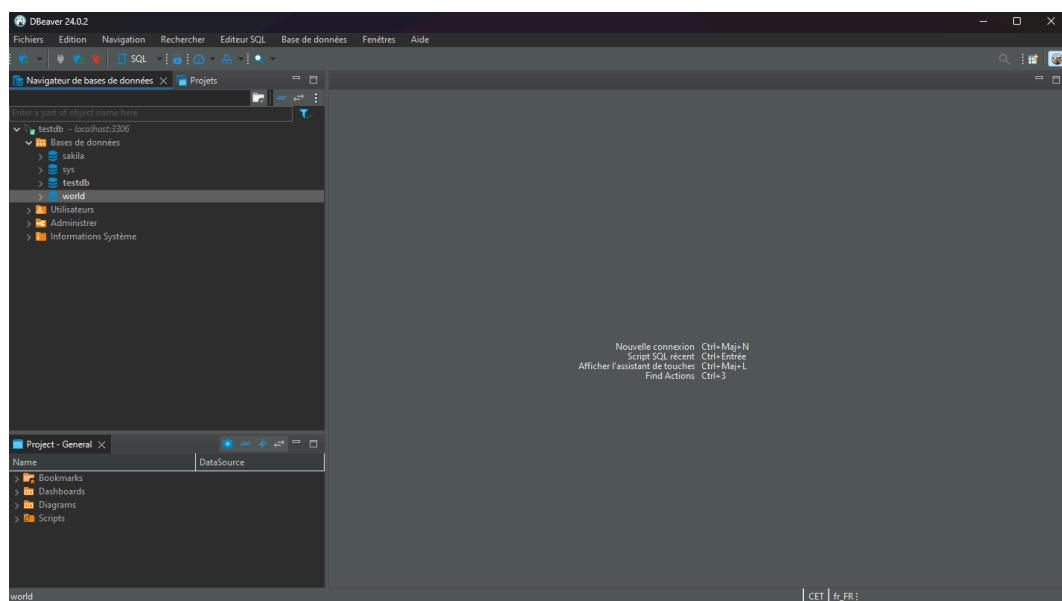
Étape 3 : Migration avec DBeaver

1. Connexion aux bases de données

Avec l'interface Dbeaver on a pu faire la connexion aux bases données :



On peut voir qu'on a bien récupéré la base de données “testdb” (MySQL) :



2. Migration

Pour débiter la migration il faut activer le service postgresql :

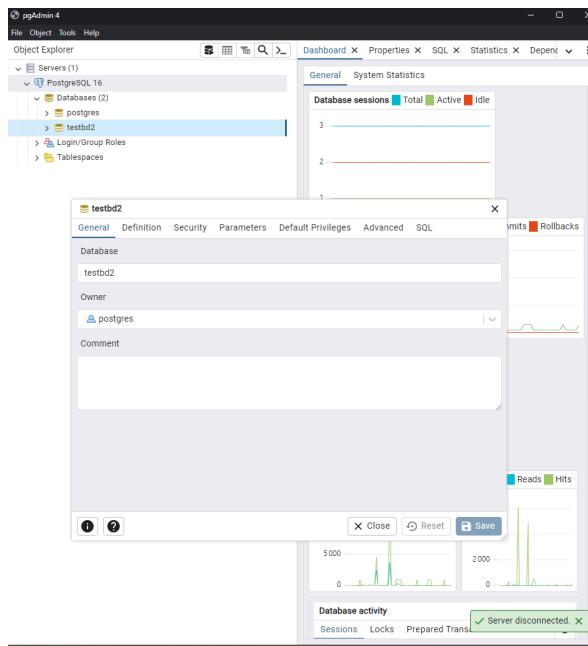
```
Administrateur : Invite de commandes
Microsoft Windows [version 10.0.22631.3447]
(c) Microsoft Corporation. Tous droits réservés.

C:\Windows\System32>net start postgresql-x64-16
Le service demandé a déjà été démarré.

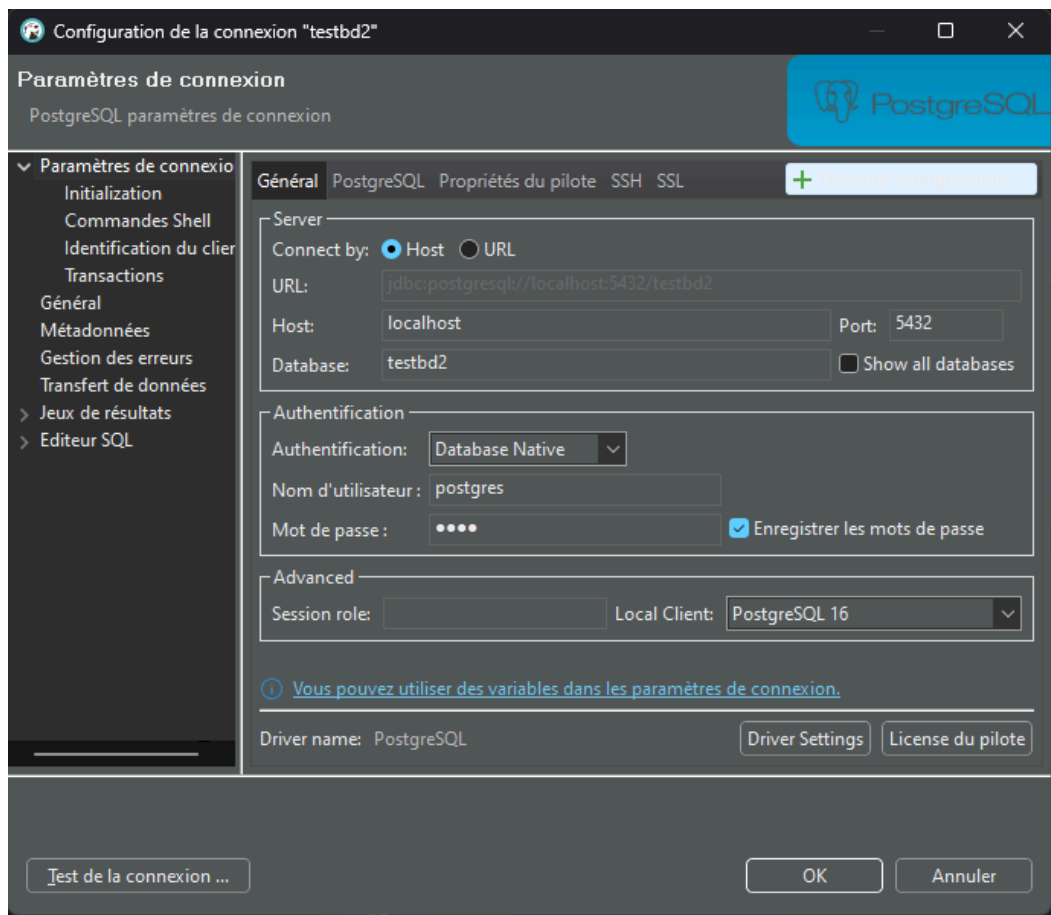
Vous obtiendrez une aide supplémentaire en entrant NET HELPMSG 2182.

C:\Windows\System32>
```

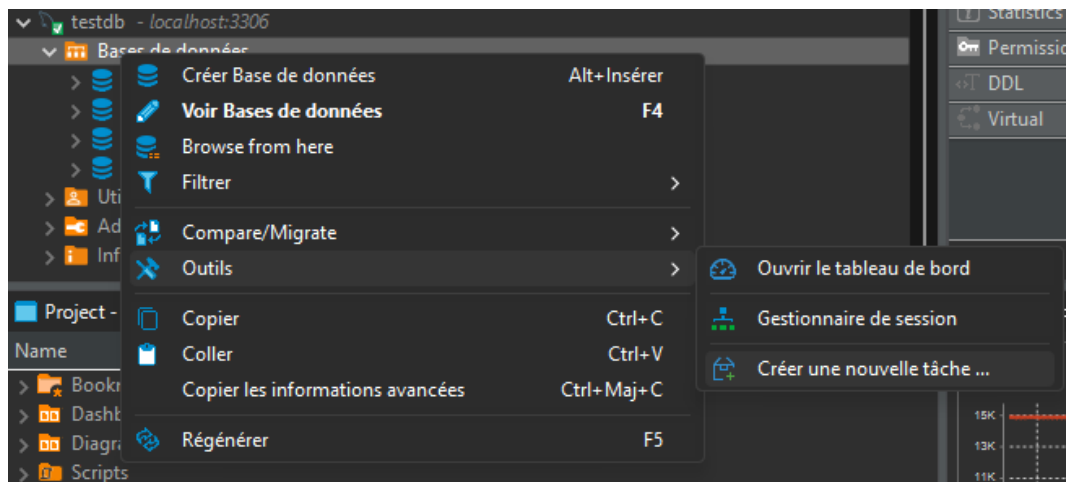
Une fois la mise en place du service effectué on peut accéder via pgadmin au tableau de bord de postgresql, où une base de donnée sera nommé “testdb2” (PostgreSQL).



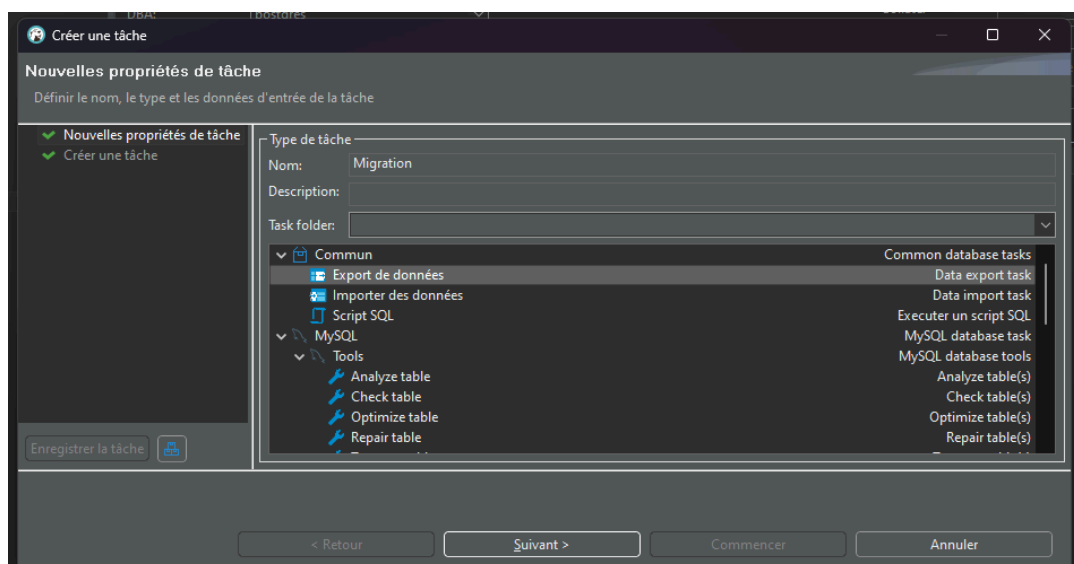
Nous devons maintenant connecté la base PostgreSQL a DBeaver comme ceci :



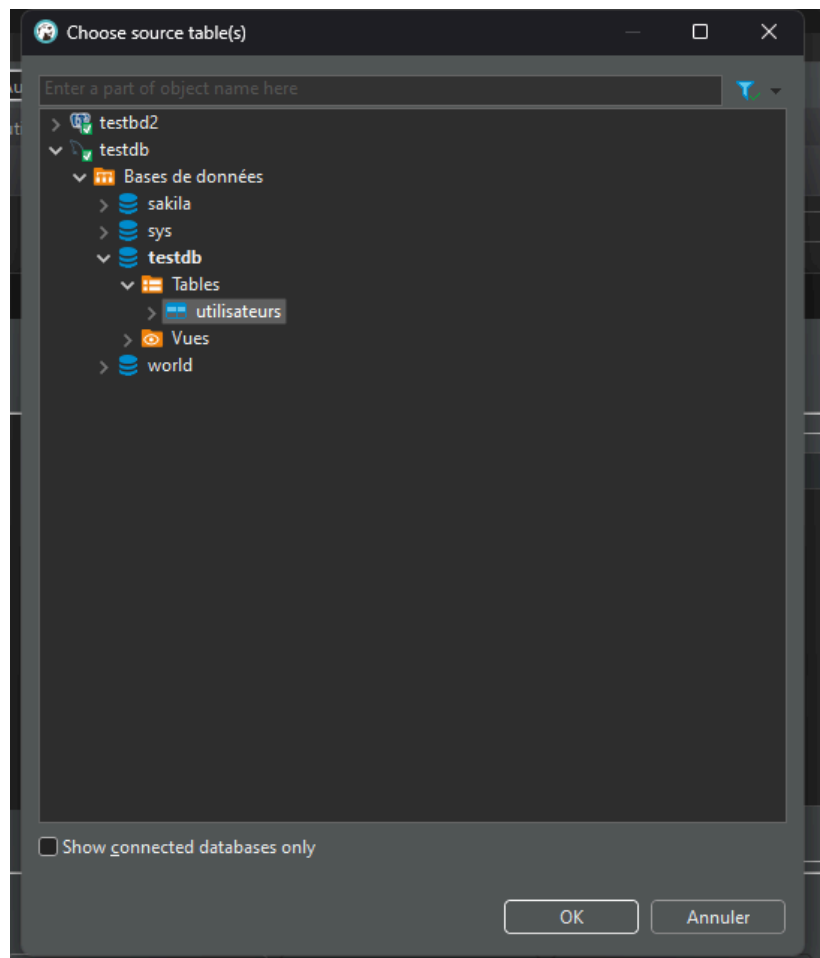
On crée également une nouvelle tâche sur “testdb” (MySQL) :



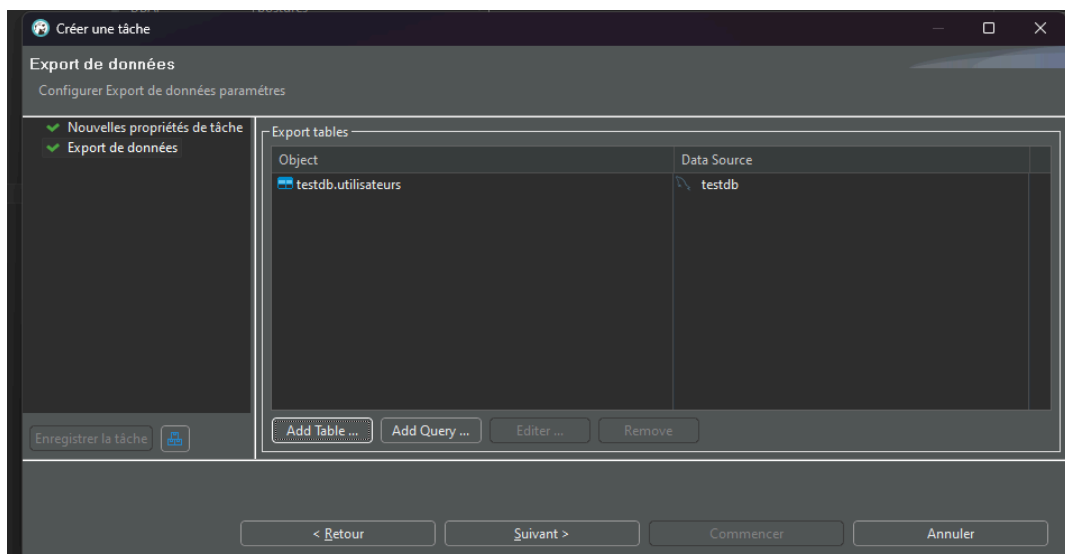
On paramètre “export de donnée” pour procéder à la migration :



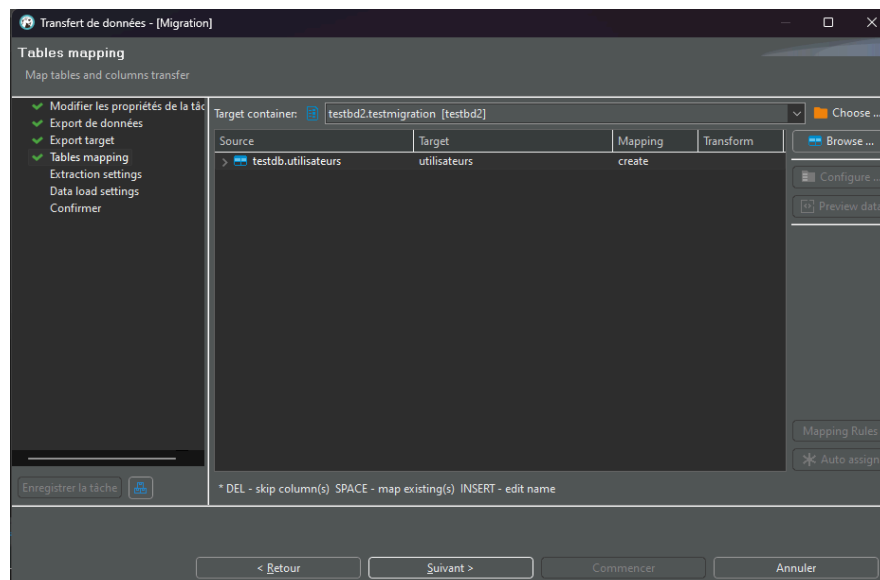
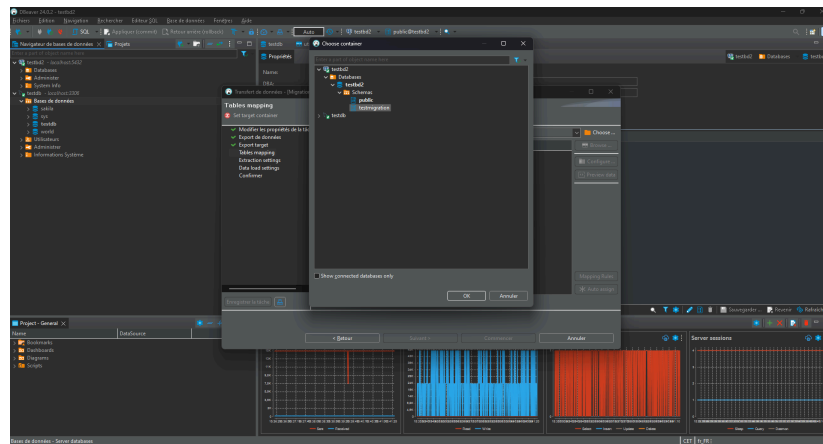
On devra sélectionner une source pour les tables :



Avec les données correspondantes liées à l'objet de la base :



On sélectionne la base de donnée PostgreSQL cible :

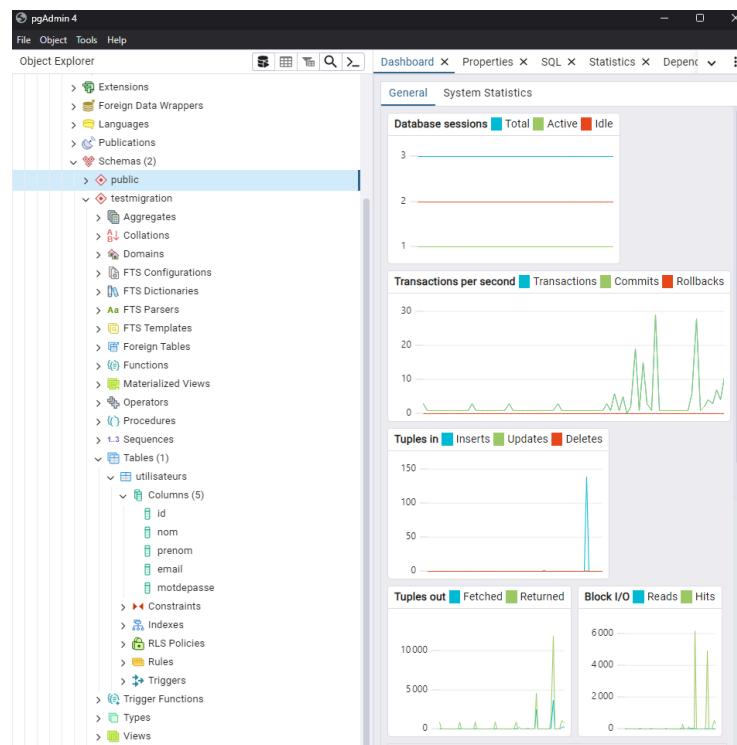


On finit par lancer la migration et nous allons donc vérifier si cela a bien fonctionné

Étape 4 : Vérification

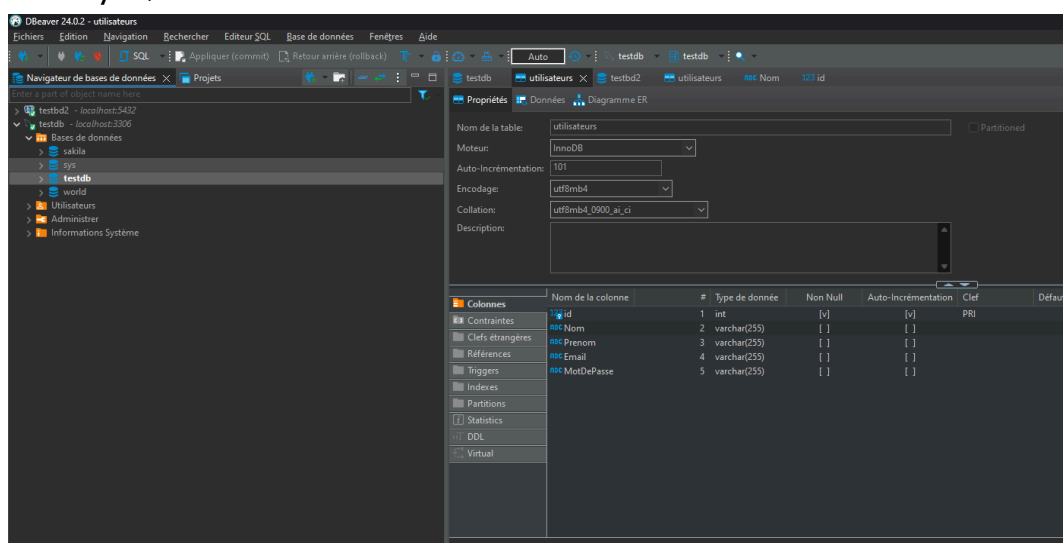
1. Vérification de la migration

Pour la migration on a pu constater que sur la Base PostgreSQL via PgAdmin on voit la migration d'effectuer car les informations concernant la base MySQL y figurent.

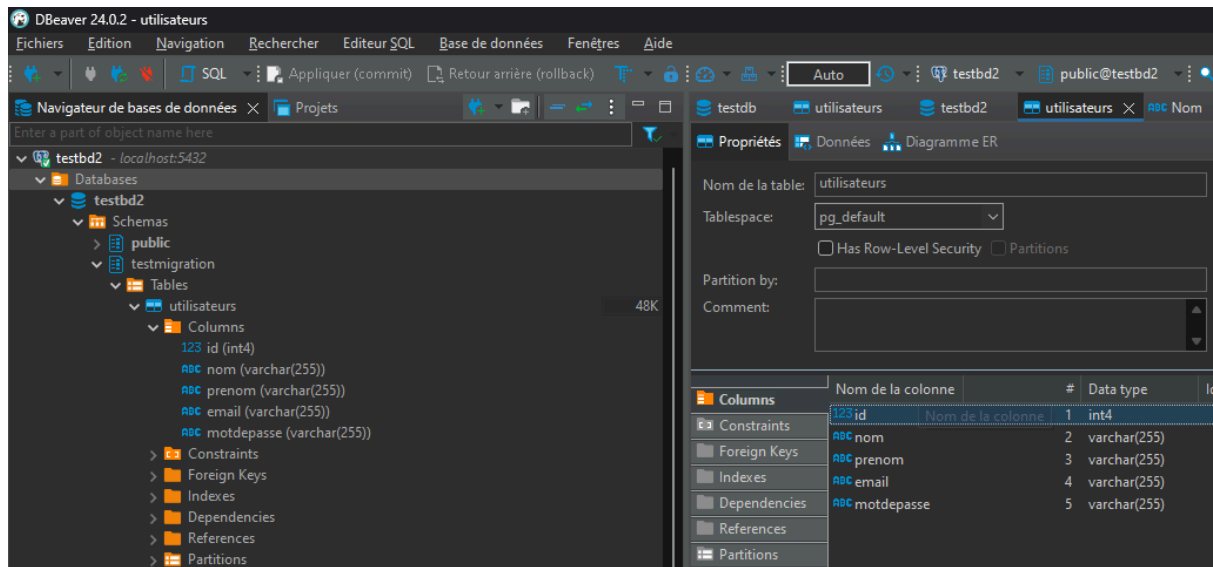


On peut également voir que sur DBeaver tout semble être bon au niveau des données :

“TestBD” MySQL :



“Testbd2” PostgreSQL :



On peut donc conclure que la base de données a bien migré.

Niveau 2 :

Étape 1 : Configuration initiale

Pour la configuration nous avons utilisé une VM debian ou l'instance MySQL et Postgre seront installés, nous avons aussi fait l'installation de flyway.

1. Configuration de l'instance MySQL

Nous avons initié la création d'une nouvelle base de données comme demandé dans l'énoncé, sous Debian nous avons pu installer et configurer celle-ci.

2. Configuration de l'instance PostgreSQL

Nous avons également initié la création de la base PostgreSQL, sous Debian nous avons pu installer et configurer celle-ci.

Étape 2 : Préparation de la base de données MySQL

1. Création de la base de donnée et insertion des donnée MySQL

Pour commencer nous avons la base de donné MySQL avec sa création :

```
MariaDB [(none)]> CREATE DATABASE testDB;  
Query OK, 1 row affected (0.000 sec)  
  
MariaDB [(none)]> USE testDB;  
Database changed
```

Nous avons fait également fait un peuplement via l'aide d'un script comme vu pour le niveau 1.

```
root@debian:~# python3 script.py  
Les tables ont été créées et les données ont été insérées avec succès dans la base de données MySQL.  
root@debian:~#
```

Nous avons donc la vision du peuplement de la BD via des requêtes ce qui nous permet d'avoir un aperçu du résultat du script.

```
MariaDB [(none)]> use testdb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [testdb]> select * from Utilisateurs;
+-----+-----+-----+-----+
| id | Nom | Prenom | Email |
+-----+-----+-----+-----+
| 1 | Wise | Robert | gblair@stafford-grant
.info |
| 2 | Olson | Glenn | ayalasydney@johnson-w
ilson.com |
| 3 | Foster | Patrick | johnsonpatricia@juare
z.com |
| 4 | Sanchez | Megan | john17@gmail.com |
| 5 | Bailey | Marc | newmanlisa@yahoo.com |
| 6 | Duncan | Bryan | zpace@taylor-smith.co
m |
| 7 | Hogan | Thomas | bobrien@hotmail.com |
| 8 | Chen | Sarah | smithangela@terry.com |
| 9 | Baldwin | Bruce | leachariel@weaver.com |
| 10 | Evans | Jason | kathleengregory@gmail
.com |
| 11 | Reyes | Edward | jeffrey05@hotmail.com |
| 12 | Lane | Nancy | jackroberts@hotmail.c
om |
| 13 | Summers | Joseph | jeremy09@tanner-burto
n.com |
| 14 | Garner | Jacqueline | erhodes@hotmail.com |
| 4cfLkTy2$g q |
```

Étape 3 : Configuration de Flyway pour la migration

1. Création de l'environnement Flyway

Pour la création de l'environnement de Flyway nous avons dû lancer des commandes d'installation fournie par le site web, le décompresser, et renseigner son chemin via le répertoire des package.

```
root@debian:~# wget -O flyway.tar.gz https://repo1.maven.org/maven2/org/flywaydb/flyway-commandline/7.0.4/flyway-commandline-7.0.4-linux-x64.tar.gz
--2024-04-21 16:55:13-- https://repo1.maven.org/maven2/org/flywaydb/flyway-commandline/7.0.4/flyway-commandline-7.0.4-linux-x64.tar.gz
Resolving repo1.maven.org (repo1.maven.org)... 199.232.196.209, 199.232.192.209, 2a04:4e42:4c::209, ...
Connecting to repo1.maven.org (repo1.maven.org)|199.232.196.209|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 95304340 (91M) [application/x-gzip]
Saving to: 'flyway.tar.gz'

flyway.tar.gz      100%[=====] 90.89M  51.2MB/s  in 1.8s
2024-04-21 16:55:16 (51.2 MB/s) - 'flyway.tar.gz' saved [95304340/95304340]
```

```
root@debian:~# tar -xzf flyway.tar.gz
root@debian:~#
```

```
root@debian:~# mv flyway.tar.gz /opt/flyway
root@debian:~# export PATH="$PATH:/opt/flyway"
```


2. Configuration de Flyway

Comme demandé dans l'énoncé la création d'un fichier de configuration qui va permettre de cibler la base de données PostgreSQL en désignant les informations de celle-ci.

```
GNU nano 5.4 flyway.conf *
flyway.url=jdbc:postgresql://localhost:5432/testdb
flyway.user=postgres
flyway.password=password
```

2. Préparation des scripts de migration

Dans les scripts de migration, on a dû créer un moyen de recréer la structure de la base de donnée MySQL puis un autre fichier avec la création des données contenant la base.

```
GNU nano 5.4 V1_creation_base_de_donnees.sql
CREATE DATABASE nom_de_la_base_de_donnees;

GNU nano 5.4 V2_migration_donnees.sql
-- Créer une table et insérer des données de test
CREATE TABLE IF NOT EXISTS utilisateur (
  id SERIAL PRIMARY KEY,
  nom VARCHAR(100),
  prenom VARCHAR(100),
  email VARCHAR(100)
);

INSERT INTO utilisateur (nom, prenom, email) VALUES
('Doe', 'John', 'john.doe@example.com'),
('Smith', 'Jane', 'jane.smith@example.com');
```

Étape 4 : Lancement de la migration avec Flyway

1. Exécution de Flyway

Pour le lancement de la migration il faut lancer la commande “flyway migrate”, comme on peut le voir ci-dessous avec les deux migration de bd.

```
root@debian:/opt/flyway/sql# flyway migrate
Flyway Community Edition 7.0.4 by Redgate
Database: jdbc:postgresql://localhost:5432/testdb (PostgreSQL 13.14)
Successfully validated 2 migrations (execution time 00:00.101s)
Current version of schema "public": << Empty Schema >>
Migrating schema "public" to version "1 - creation base de donnees" [non-transactional]
Migrating schema "public" to version "2 - migration donnees"
Successfully applied 2 migrations to schema "public" (execution time 00:00.274s)
root@debian:/opt/flyway/sql#
```

Étape 5 : Vérification de la migration

1. Vérification dans PostgreSQL

Pour la vérification de la migration il faut aller consulter sur la bd PostgreSQL via des requêtes et on peut constater que la table utilisateur y est bien renseigné.

```
postgres=# \c testdb
You are now connected to database "testdb" as user "postgres".
testdb=# \dt

          List of relations
Schema |          Name          | Type  | Owner
-----+-----+-----+-----
public | flyway_schema_history | table | postgres
public | utilisateur            | table | postgres
(2 rows)
```

On peut voir que des utilisateur généré précédemment y figure.

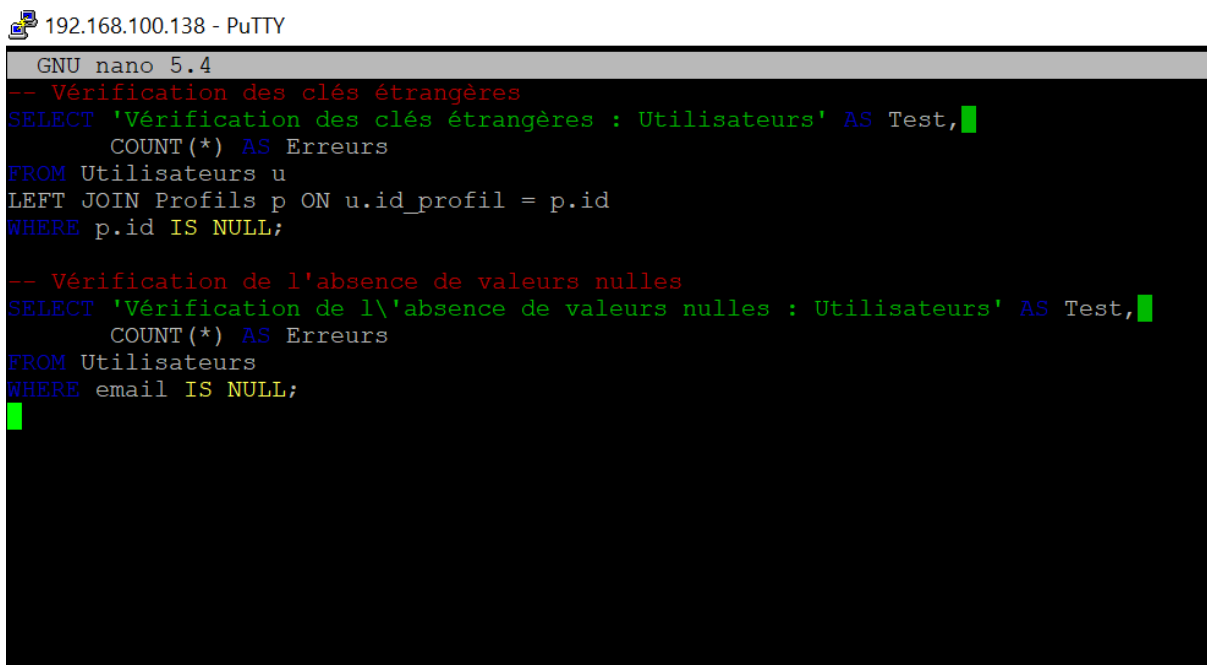
```
testdb=# select * from utilisateur;
 id | nom  | prenom |          email
----+-----+-----+-----
  1 | Doe  | John   | john.doe@example.com
  2 | Smith | Jane   | jane.smith@example.com
(2 rows)
```

Niveau 3

Étape 1 : Créer des scripts de test

V3__Test_integrite :

Ces requêtes permettent de vérifier si les données dans la table Utilisateurs sont conformes à certaines règles d'intégrité, comme l'existence de clés étrangères valides et l'absence de valeurs nulles dans certaines colonnes importantes. Ces vérifications sont cruciales pour s'assurer que la migration des données s'est déroulée correctement et que la base de données est cohérente après la migration.

A screenshot of a terminal window titled '192.168.100.138 - PuTTY'. The terminal shows the GNU nano 5.4 editor with two SQL queries. The first query is for verifying foreign keys, and the second is for verifying the absence of null values in the email column. Both queries use the 'Utilisateurs' table and join it with the 'Profils' table.

```
GNU nano 5.4
-- Vérification des clés étrangères
SELECT 'Vérification des clés étrangères : Utilisateurs' AS Test,
       COUNT(*) AS Erreurs
FROM Utilisateurs u
LEFT JOIN Profils p ON u.id_profil = p.id
WHERE p.id IS NULL;

-- Vérification de l'absence de valeurs nulles
SELECT 'Vérification de l\'absence de valeurs nulles : Utilisateurs' AS Test,
       COUNT(*) AS Erreurs
FROM Utilisateurs
WHERE email IS NULL;
```

V4__Test_completude :

Dans ce script, nous vérifions le nombre d'utilisateurs et si chaque utilisateur a un profil correspondant. Nous vérifions également le nombre total d'enregistrements dans les tables Utilisateurs et Profils.

```
192.168.100.138 - PuTTY
GNU nano 5.4 V4_Test_completude.sql *
-- Vérification de la complétude des données dans Utilisateurs
SELECT 'Vérification de la complétude des données : Utilisateurs' AS Test,
COUNT(*) AS Erreurs
FROM Utilisateurs;

-- Vérification du nombre total d'enregistrements
SELECT 'Vérification du nombre total d'enregistrements dans toutes les tables' AS Test,
COUNT(*) AS Total
FROM (SELECT * FROM Utilisateurs UNION ALL
SELECT * FROM Profils) AS AllTables;
```

Étape 2 : Exécuter Flyway

Ce script exécute Flyway avec les scripts SQL de test et vérifie le code de sortie de l'exécution. S'il est égal à 0, cela signifie que les tests ont été exécutés avec succès, sinon, une erreur est affichée et le script se termine avec un code de sortie différent de zéro.

```
192.168.100.138 - PuTTY
GNU nano 5.4 run_tests.sh *
#!/bin/bash

# Chemin absolu vers le dossier contenant les scripts SQL de test
SQL_DIR=$(pwd)/sql

# Chemin absolu vers le dossier contenant le fichier de configuration Flyway
CONF_DIR=$(pwd)/conf

# Nom de l'image Docker Flyway
FLYWAY_IMAGE=flyway/flyway

# Commande pour exécuter Flyway avec les scripts de test
echo "Exécution des scripts de test avec Flyway..."
docker run --rm -v $SQL_DIR:/flyway/sql -v $CONF_DIR:/flyway/conf $FLYWAY_IMAGE migrate

# Vérification du résultat de l'exécution
if [ $? -eq 0 ]; then
    echo "Les tests ont été exécutés avec succès."
else
    echo "Erreur lors de l'exécution des tests."
    exit 1
fi
```