

CYCLE INITIAL EN TECHNOLOGIES DE L'INFORMATION DE SAINT-ÉTIENNE

TP RCP30


Base de Données

LUCAS LESCURE



Table of Content

1. D'emarrage de PostgreSQL	3
2. Construction de la Base de données	3
2.1. Phase 1: importation du dump dans la base de données.	4
2.2. Phase 2 : création des tables type_plateforme et contenu_genre.....	4
2.3. Phase 3: Insertion et mise à jour de donnés.	5



1. D'emarrage de PostgreSQL

On récupère sous Moodle le kit `KitEtuBddPgSQL.zip` que l'on procède par extraire, ainsi que tout les fichiers .zip contenu à l'intérieur.

En ouvrant alors une fenêtre de commande on se redirige à l'emplacement de ce fichier et on renomme le répertoire `pgsql1` à `PostgreSQL_12.4_Portable`. Ensuite on commence par créer un nouveau **cluster** de base de données en modifiant le fichier `Make Cluster.bat` avec la commande suivante :

```
"%CD%\bin\initdb" -D "%RutaCluster%" -U postgres -W --encoding=UTF8
```

On peut alors lancer le script dans la ligne de commande avec `"Make Cluster.bat"`. L'exécution du fichier demande un nom au répertoire du nouveau cluster. On le nommera `data2`. Une fois crée entrer le mot de passe de l'administrateur de la base de donnée.

On commence désormais par modifier les fichiers `PostgreSQL-Start.bat` `PostgreSQL-Restart.bat` `PostgreSQL-Stop.bat` en plaçant chacun des `data` en `data2`, exemple:

```
"%CD%\bin\pg_ctl.exe" start -D "%CD%\data2"
```

Dans la command line on peut donc commencer le serveur SQL en tapant la commande `"PostgreSQL-Start.bat"`. Une fois initialisé on peut la modifier en utilisant le logiciel `pgAdmin`

Dans le client `pgAdmin` on crée une nouvelle connexion au serveur de façon suivante:

- nom: `PostgreSQL_12.4_Portable`
- hôte: `localhost`
- port: `5432`
- base maintenance: `postgres`
- username: `postgres`
- password: <Mot de passe>

Après ceci on ouvre le `Query Tool` et on crée notre serveur à partir de la commande :

```
CREATE DATABASE plateformes_multimedia OWNER postgres ;
```

2. Construction de la Base de données

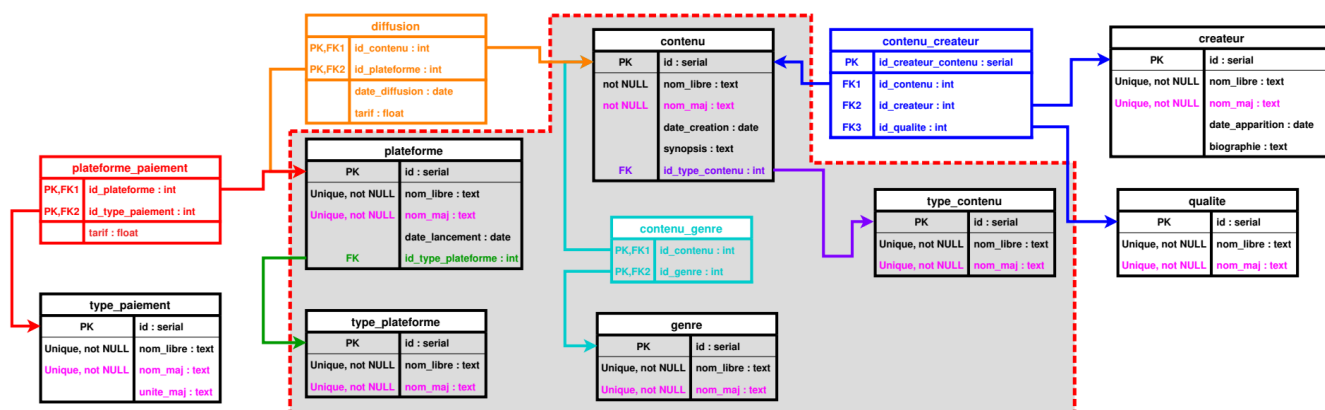


Figure 2.1. Modèle relationnel à réaliser

2.1. Phase 1: importation du dump dans la base de données

Une partie est déjà créée dans le fichier `dump_plateformes_multimedia.sql`. C'est ce que l'on appelle un dump, et contient l'ensemble des requêtes SQL qui ont permis de créer et remplir une partie de la base.

On procède par ouvrir ce fichier en étant sur le Query Tool puis on exécute (avec F5). À la suite on supprime tout le code et on tape la commande:

```
set search_path = 'public' ;
```

2.2. Phase 2 : création des tables `type_plateforme` et `contenu_genre`

Après l'analyse des éléments on remarque qu'il manque la table `type_plateforme` et la table tampon `contenu_genre`. On va donc devoir les créer tout seul en respectant les contraintes associés.

Concernant la table `type_plateforme` il faudra, une fois créée, exécuter le script `trigger_insert_update_type_plateforme`. On devra ensuite ajouter l'attribut `id_type_plateforme` dans la table `plateforme` et poser la contrainte de clé étrangère vers l'id de `type_plateforme`.

Pour ce qui concerne le dump, on peut se référer à la phase 1 pour se faire.

Pour la création de la table on peut utiliser la commande suivante :

```
CREATE TABLE type_plateforme(ID serial PRIMARY KEY,
                               nom_libre text UNIQUE NOT NULL,
                               nom_maj text UNIQUE NOT NULL);
```

Pour ensuite configurer la clé étrangère on doit d'abord créer l'attribut auquel appartiendra la clé étrangère pour ceci on peut se référer au code :

```
ALTER TABLE plateforme ADD COLUMN id_type_plateforme int;
```

Ensuite on configure la clé étrangère avec la commande :

```
ALTER TABLE plateforme ADD CONSTRAINT fk_id_type_plateforme
FOREIGN KEY (id_type_plateforme) REFERENCES type_plateforme(ID)
ON DELETE SET NULL
ON UPDATE CASCADE;
```

Dans lequel on référence la clé étrangère directement au tableau associé `type_plateforme` comme il est décrit dans le modèle relationnel.

On répète ceci pour la table `contenu`.

On réalise en utilisant donc ce principe de clé la table tampon suivante:

```
CREATE TABLE contenu_genre ()
ALTER TABLE contenu_genre ADD COLUMN id_contenu int;
ALTER TABLE contenu_genre ADD COLUMN id_genre int;

ALTER TABLE contenu_genre ADD CONSTRAINT pk_contenu
PRIMARY KEY (id_contenu, id_genre);

ALTER TABLE contenu_genre ADD CONSTRAINT fk_id_contenu
FOREIGN KEY (id_contenu) REFERENCES contenu(ID)
ON DELETE CASCADE
ON UPDATE CASCADE;

ALTER TABLE contenu_genre ADD CONSTRAINT fk_id_genre
FOREIGN KEY (id_genre) REFERENCES genre(ID)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

2.3. Phase 3: Insertion et mise à jour de donnés

Dans cette phase on veut pouvoir faire des requête d'insertion (INSERT INTO), de suppression (DELETE) et de mise à jour (UPDATE).

Avec un fichier excel, data.xlsx on se propose de transférer ces donnés dans notre base de données. Pour la copie directe de donnés, il faut sauvegarder notre excel sous format .csv et veiller à ce que le codage soit en UTF-8. Ensuite on peut alors copier ses donnés en suivant la commande :

```
COPY type_plateforme (ID,nom_libre)
FROM 'D:\Users\lucas\Downloads\KitEtuBddPgSQL\KitEtuBddPgSQL' CSV
DELIMITER ',';
```