Secteur Tertiaire Informatique

Filière « Etude et développement »

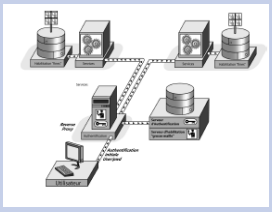
TP – Gestion de commandes

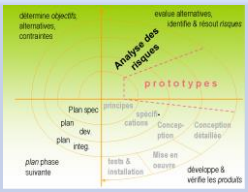
Persistance de données

Apprentissage

Mise en situation

Evaluation





1. MISE EN PLACE

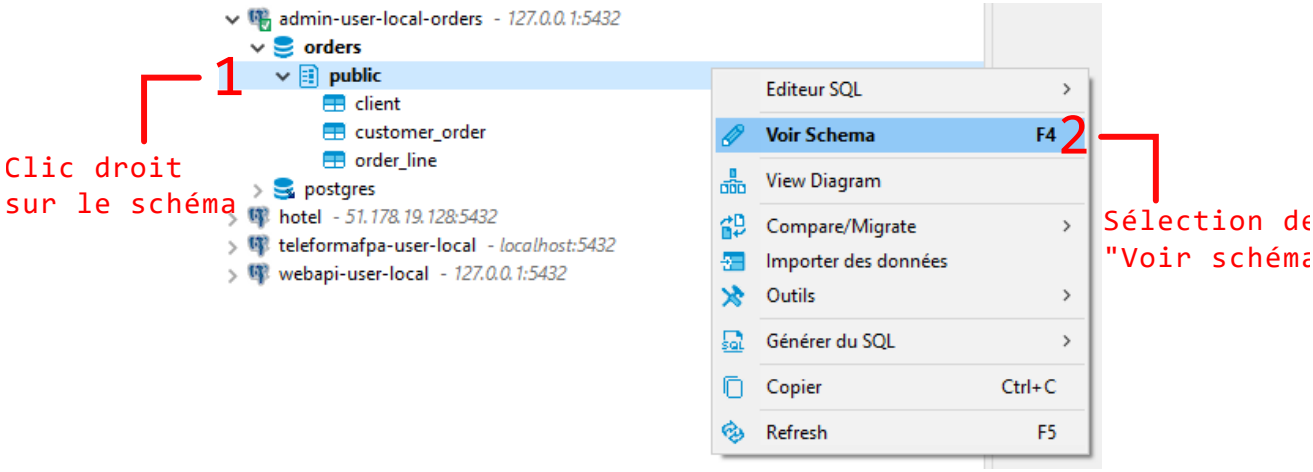
Récupérez le script de création de base de données disponible via Git (hébergé sur Gitlab) et déployez en local une nouvelle base de données appelée « orders ».

Il vous faudra ajouter le « Dockerfile » à utiliser pour pouvoir instancier une image.

1.1 DIAGRAMME D'ENTITE/RELATION

Avant toute chose, il est intéressant d'observer la structure de la base de données grâce à une représentation graphique des tables.

Un outil tel que DBeaver permet d'éditer un diagramme nommé « Entité/Relation » (équivalent du « modèle logique de données » de la méthode Merise).



Etudiez le diagramme d'entité-relation et **répondez aux questions suivantes** (vos réponses pourront être notées dans un script sql comprenant également les requêtes à écrire, vous pourrez commenter ces réponses) :

- A correspond la table « order_line » ?
- A quoi sert la colonne « order_id » ?

Une fois la base en place et la structure comprise, vous pourrez écrire les requêtes répondant aux besoins présentés par la partie 2 de ce document.

2. REQUETES A ECRIRE

1. **Récupérer l'utilisateur ayant le prénom "Muriel" et le mot de passe "test11", sachant que l'encodage du mot de passe est effectué avec l'algorithme Sha1.**

L'idée est ici de simuler une requête qui pourrait être effectuée lors de la connexion d'un utilisateur. L'utilisateur saisit son mot de passe en clair qui sera hashé en utilisant Sha1.

Il vous faudra faire appel à une fonction de hashage en SQL pour transformer le mot de passe « test11 ».

Pour des raisons de sécurité il est **INTERDIT de stocker les mots de passe en clair en base de données**. Il faudra utiliser une fonction de hachage permettant de stocker une empreinte cryptographique du mot de passe.

Je vous conseille de visionner la vidéo suivante pour mieux comprendre l'idée derrière une fonction de hachage : https://www.youtube.com/watch?v=OHXfKCH0b6s

Pour chiffrer une chaîne de caractères en utilisant PostgreSQL vous pourrez utiliser la fonction « digest(data, type) » qui retrouve le « hash » d'une chaîne de caractères suivant un algorithme défini.

La fonction « digest » prend deux paramètres :

- « **data** » : correspondant à la **chaîne de caractères à traiter**
- « **type** » : chaîne de caractères qui indique le type d'algorithme de hachage à utiliser. Les algorithmes pouvant être utilisés sont : `md5`, `sha1`, `sha224`, `sha256`, `sha384` et `sha512`.

« digest() » renvoie un « hash » en sous la forme d'une chaîne binaire (c'est le type « bytea » pour PostgreSQL).

Pour obtenir la forme hexadécimale que l'on souhaite stocker en base de données il vous est possible d'utiliser la fonction « encode(data, type) ».

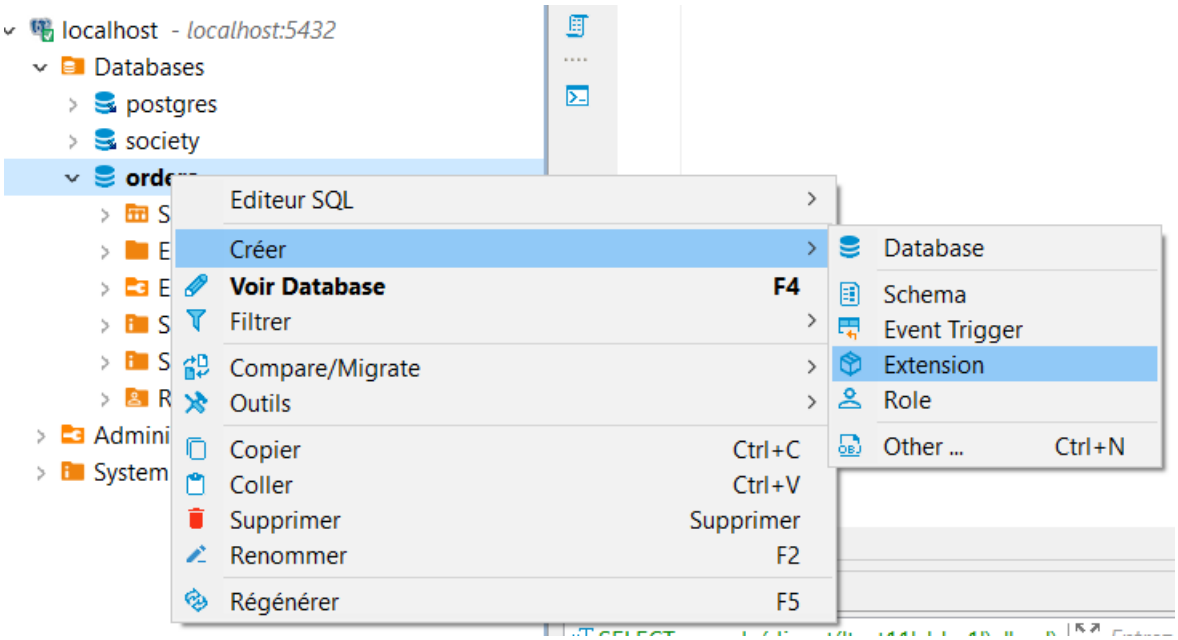
Par exemple, pour retrouver le hash de la chaîne « motdepassesupersecure » en utilisant l'algorithme SHA1, vous pourrez utiliser :

```
SELECT encode(digest('motdepassesupersecure', 'sha1'), 'hex');
```

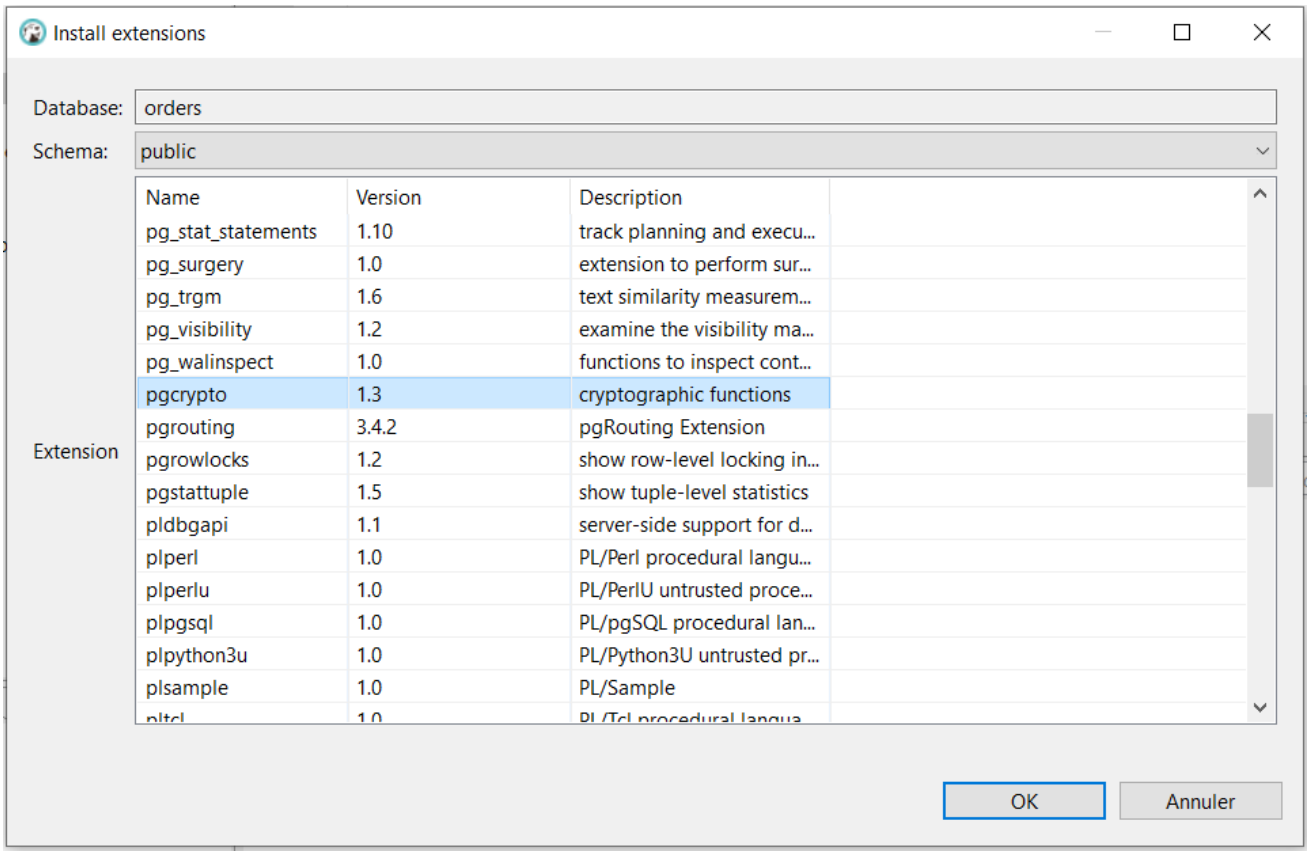
Une fonction peut être appelée après un « SELECT » ou après une clause « WHERE ».

Afin de pouvoir utiliser les fonctions de crypthographies en PostgreSQL il est nécessaire d'activer une extension du SGBD : **pgcrypto.**

Sur DBeaver vous pouvez faire un clic droit sur votre base de données « **orders** », dans le menu contextuel qui s'affiche vous retrouverez « Créer » puis « Extension ».



Une fois la fenêtre d'installation d'extension affichée, sélectionnez « **pgcrypto** » pour votre schéma public.



2. **Récupérer la liste de tous les produits qui sont présents sur plusieurs commandes.**

3. **Enregistrer le prix total à l'intérieur de chaque ligne des commandes, en fonction du prix unitaire et de la quantité (il vous faudra utiliser une requête de mise à jour d'une table : « UPDATE TABLE », vous pourrez vous inspirer de la première réponse de cette questions Stackoverflow).**

Pour plus d'informations concernant le fonctionnement de « UPDATE TABLE » vous pouvez vous référer à la ressource suivante : **https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-update/**

4. **Récupérer le montant total pour chaque commande et afficher la date de commande ainsi que le nom et le prénom du client.**

5. **Récupérer le montant global de toutes les commandes, pour chaque mois.**

   Vous pourrez extraire le numéro du mois en utilisant la méthode « extract(field from source) » de PostgreSQL.
   Le résultat obtenu pourra être utilisé dans un « **GROUP BY** ».

6. **Récupérer la liste des 10 clients qui ont effectué le plus grand montant de commandes, et obtenir ce montant total pour chaque client.**

7. **Récupérer le montant total des commandes pour chaque jour.**

8. **Ajouter une colonne intitulée "category" à la table contenant les commandes. Cette colonne contiendra une valeur numérique (il faudra utiliser « ALTER TABLE »,** https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-add-column/**)**

9. **Enregistrer la valeur de la catégorie, en suivant les règles suivantes :**
   - **"1" pour les commandes de moins de 200€**
   - **"2" pour les commandes entre 200€ et 500€**
   - **"3" pour les commandes entre 500€ et 1.000€**
   - **"4" pour les commandes supérieures à 1.000€**

Plusieurs solutions peuvent être envisagées pour répondre à la demande :

- Utiliser plusieurs requête « UPDATE » en adaptant la condition ;
- Utiliser une seule requête « UPDATE » en y ajoutant un « CASE ».

Voici un exemple d'utilisation d'un « UPDATE » avec un « CASE » :

```
UPDATE <nom-table>
SET <nom-colonne-à-mettre-à-jour> = CASE
  WHEN <nom-colonne-à-tester> THEN <valeur-1>
  WHEN <nom-colonne-à-tester> THEN <valeur-2>
  ELSE <valeur-défaut>
END;
```

Imaginons une table « employee » avec une colonne « salary », on pourrait imaginer mettre le salaire d'un employé à jour en fonction de son service (« department ») avec la requête suivante :

```
UPDATE employee
SET salary = CASE
 WHEN department_id = 1 THEN salary * 10
 WHEN department_id = 13 THEN salary / 2
 ELSE salary
END;
```

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
Michel Coulard – Formateur Evry
Chantal Perrachon – IF Neuilly sur Marne

Date de mise à jour : 10/09/2023