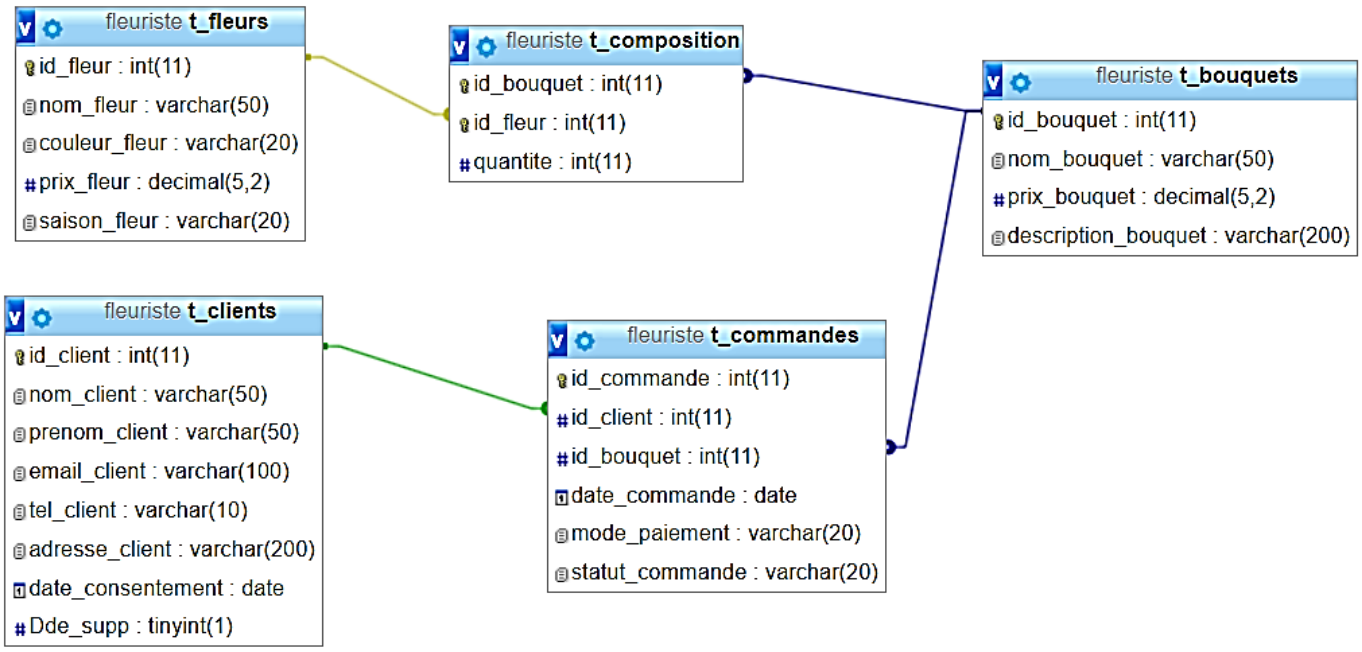
B3.1 : Cybersécurité - Data

TP6 – L'accès aux données - BDD

Lundi 2 mai

Soit le schéma Mysql de la base de données du fleuriste "Le Bouquet Parfait"



Vous disposez d'un script sql de création de cette bdd, de ses tables et d'insertion d'un jeu de données.

PARTIE 1

1) Respecter le souhait de confidentialité de certains clients
   - Pseudomiser des informations
   - Anonymiser des informations
2) Nettoyer la bdd (archiver des données)

PARTIE 2

On souhaite différencier 3 types d'utilisateur de cette bdd :

- Le Fleuriste et le responsable de la boutique et les client.

Chacun d'eux n'ont pas le même intérêt et le même usage

3) Créer des vues spécifiques à chaque utilisateur
4) Créer des compte utilisateurs

PARTIE 1

1 - La confidentialité des DCP

Travail demandé :

Proposer 2 méthodes pour chaque principe = anonymiser et pseudomiser des noms des clients dont la valeur du champ dde_supp est à vrai

Effectuer 4 copies de la table clients, faire chaque test sur une copie différente.

Décrire vos propositions (méthode utilisée - code sql) et faire une critique :avantage et inconvénients) :

Anonymiser

	Solution 1	Solution 2
Méthode Utilisée		
Script SQL		
Résultat		
Critique		

pseudomiser

	Solution 1	Solution 2
Méthode Utilisée		
Script SQL		
Résultat		
Critique		

ANNEXE 1 : Fonctions SQL utiles

- **RAND()** : génére une valeur aléatoire.

Syntaxe :	Exemple :
`RAND()`	`SELECT RAND() AS result;` >> Résultat : Un nombre aléatoire entre 0 et 1.

Cette technique peut être utilisée pour pseudonymiser des données sensibles telles que les numéros de téléphone ou les numéros de sécurité sociale.

- **FLOOR** : Cette fonction est utilisée pour arrondir un nombre à l'entier inférieur le plus proche.

Syntaxe :	Exemple :
`FLOOR(number)`	`SELECT FLOOR(3.7) AS result;` >> Résultat : 3.

- **CONCAT()** : concaténe plusieurs chaînes de caractères.

Syntaxe :	Exemple :
`CONCAT(string1, string2, ...)`	`SELECT CONCAT('Hello', ' ', 'World') AS result;` >> Résultat : 'Hello World'

Cette technique peut être utilisée pour pseudonymiser des données sensibles en combinant les différentes parties du nom ou de l'adresse.

- **REPLACE()** : remplace une chaîne de caractères par une autre.

Cette technique peut être utilisée pour remplacer les noms ou les adresses par des valeurs génériques telles que "John Doe" ou "123 Main Street".

Syntaxe :	Exemple :
`REPLACE(string, search_value, replacement_value)`	`SELECT REPLACE('Hello, World!', ',', '-') AS result;` >> Résultat : 'Hello- World!'

- **REPEAT** : Cette fonction est utilisée pour répéter une chaîne de caractères un certain nombre de fois.

Syntaxe :	Exemple :
`REPEAT(string, number_of_times)`	`SELECT REPEAT('Hello', 3) AS result;` >> Résultat : 'HelloHelloHello'

- **SUBSTRING** : Cette fonction est utilisée pour extraire une partie d'une chaîne de caractères.

Syntaxe :	Exemple :
`SUBSTRING(string, start, length)`	`SELECT SUBSTRING('Hello World', 1, 5) AS result;` >> Résultat : 'Hello'

LEFT : Cette fonction est utilisée pour extraire les premiers caractères d'une chaîne de caractères.

Syntaxe :	Exemple :

`LEFT(string, length)`	`SELECT LEFT('Hello World', 5) AS result;` >> Résultat : 'Hello'

RIGHT : Cette fonction est utilisée pour extraire les derniers caractères d'une chaîne de caractères.

Syntaxe :	**Exemple :**
`RIGHT(string, length)`	`SELECT RIGHT('Hello World', 5) AS result;` >> Résultat : 'World'

- **Fonctions de hachage**
  - **MD5()** transforme une chaîne de caractères en une valeur de hachage.

Syntaxe :	**Exemple :**
`MD5(string)`	`SELECT MD5('Hello World') AS result;` >> Résultat : 'ed076287532e86365e841e92bfc50d8c'

Le hachage MD5 est une empreinte numérique de 128 bits généralement représentée sous forme de chaîne hexadécimale de 32 caractères.

- Plus récentes et plus robustes SHA1 et SHA2

SHA-1 est une empreinte numérique de 160 bits généralement représentée sous forme de chaîne hexadécimale de 40 caractères.

SHA2 permet de spécifier la longueur du hachage (224,256,384,512)

`SHA2(string, hash_length)`

PARTIE 2

1 - Les vues

Définition :

Une vue est représentation virtuelle d'une partie des données d'une ou plusieurs tables.

Elle est créée en utilisant une requête SQL pour extraire et afficher les données spécifiques.

Dans notre cas les vues sont permettre de produire :

- Une présentation personnalisée des données.
- De contrôler l'accès aux données en limitant les colonnes ou les lignes visibles pour certains utilisateurs.

Syntaxe :	**Exemple :**

`CREATE VIEW nom_de_la_vue AS`
`SELECT colonne1, colonne2, ...`
`FROM nom_de_la_table`
`WHERE condition;`
	`CREATE VIEW vue_clients AS`
	`SELECT nom, email`
	`FROM clients`
	`WHERE id_client > 100;`

Travail demandé : Créer Les vues suivantes :

1) V_client : qui donnent aux clients la liste des bouquets proposés par la boutique et le nombre de fleurs, le prix du bouquet….
2) V_Composition : vue destinée au fleuriste qui lui donne la composition des bouquets avec le détail des fleurs le composant (couleur, quantité) …
3) V_Vente_mois : vue pour le responsable qui lui donne pour chaque mois le montant des ventes
4) V_cout_bouquet : vue pour le responsable qui lui donne pour chaque bouquet le prix de vente et le prix coutant (calculé en fonction du prix de chaque fleur)

2- Les droits d'accès

La création d'utilisateur dans une base de données permet :

De contrôler finement les droits d'accès aux objets de la bases de données, tables et fonctionnalités

De garantir la sécurité et la confidentialité des données en limitant l'accès uniquement aux utilisateurs autorisés

Syntaxe :	**Exemple :**
`CREATE USER 'nom_utilisateur'@'hôte' IDENTIFIED BY 'mot_de_passe';`	`CREATE USER 'john'@'localhost' IDENTIFIED BY 'motdepasse123';`

Travail demandé : Créer les 3 utilisateurs suivantes :

- Fleuriste - responsable - client.

Le droit d'accès à une table en MySQL est le privilège qui permet à un utilisateur de lire, écrire, modifier ou supprimer des données dans une table.
Il existe plusieurs types de droits d'accès, comme SELECT, INSERT, UPDATE, DELETE, etc.

Pour attribuer ou révoquer des droits d'accès à un utilisateur sur une table, on utilise les commandes GRANT et REVOKE.

Syntaxe :	**Exemples :**

`GRANT privilèges ON objet TO 'nom_utilisateur'@'hôte';`
 	
Accorde tous les privilèges
`GRANT ALL PRIVILEGES ON nom_de_la_base_de_données.* TO 'nom_utilisateur'@'localhost';`
 	
Accorde des privilèges spécifiques sur une table à un utilisateur
`GRANT SELECT, INSERT, UPDATE ON nom_base_de_données.nom_table TO 'nom_utilisateur'@'localhost';`
 	
Maj des privilèges
`FLUSH PRIVILEGES;`
	`GRANT ALL PRIVILEGES ON *.* TO 'nom_utilisateur'@'localhost';`
	
	`GRANT SELECT, INSERT, UPDATE, DELETE ON nom_de_la_base_de_données.* TO 'nom_utilisateur'@'localhost';`
	
	`GRANT SELECT (colonne1, colonne2) ON nom_de_la_base_de_données.nom_de_la_table TO 'nom_utilisateur'@'localhost';`

La commande pour vérifier les droits d'un utilisateur
`SHOW GRANTS FOR 'client'@'localhost';`

Travail demandé : Compléter le tableau suivant pour définir les droits des 3 utilisateurs :

OBJETS	**Fleuriste**	**Client**	**Responsable**
T_fleurs			
V_client			
V_composition			

Utiliser l'acronyme CRUD (Create, Read, Update, Delete) ou
CIMS (Consultation, Insertion, Modif, suppresion) pour désigner les quatre opérations de base sur les données d'une table.

Travail demandé :

Allouer les droits à l'utilisateur client en ne lui allouant que le SELECT sur sa vue

Se connecter à la bdd avec l'utilisateur client et vérifier ses droits