

CRUD commande BDD

SHOW nomDeLaBase	Voir la BDD
USE nomDeLaBase	Utilisé une BDD
CREATE DATABASE nomDeLaBase	Créer une BDD
DROP DATABASE ma_base	Supprimer une BDD

Ajout d'une colonne à une table

```
ALTER TABLE nom_table
ADD nom_colonne type_donnees
```

```
EX : ALTER TABLE utilisateur
ADD adresse_rue VARCHAR(255)
```

Supprimer une colonne d'une table

```
ALTER TABLE nom_table
DROP nom_colonne
```

```
EX: ALTER TABLE animal
DROP adresse_rue
```

Renommer une colonne d'une table

```
ALTER TABLE nom_table
CHANGE colonne_ancien_nom colonne_n-
ouveau_nom type_donnees
```

```
ALTER TABLE animal
CHANGE image image DECIMAL(255)
NOT NULL;
```

Règle et convention

SQL exécute les actions de cette façon :

```
FROM
WHERE
SELECT
```

Le noms des tables doivent être en minuscules et au singulier.

Toutes les commandes sql sont en majuscule

Bonnes pratiques

Utilisez des identifiants et noms cohérents et descriptifs.

Faites un usage judicieux de l'espace et de l'indentation afin de faciliter la lecture du code.

Utilisez la norme ISO 8601 pour les informations temporelles (YYYY-MM-DDTHH:MM:SS.SSSSS).

Essayez de n'utiliser que des fonctions SQL standard au lieu des fonctions spécifiques à chaque SGBD pour des raisons de portabilité.

Gardez le code court et évitez les redondances comme les parenthèses ou guillemets superflus, ou encore les clauses WHERE qui peuvent être dérivées.

Commentez le code SQL lorsque c'est nécessaire. Utilisez le style de commentaires du langage C, en ouvrant avec */ et fermant avec /* là où c'est possible, autrement débutez les commentaires avec *-* et faites-les suivre par une nouvelle ligne.

Having

```
SELECT colonne1, SUM(colonne2)
FROM nom_table
GROUP BY colonne1
HAVING fonction(colonne2) operateur
valeur
```

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM(), COUNT(), AVG(), MIN() ou MAX().

```
SELECT client, SUM(tarif)
FROM achat
GROUP BY client
HAVING SUM(tarif) > 40
```

Jointure

```
SELECT *
FROM table1
INNER JOIN table2 ON table1.id =
table2.fk_id
OU
SELECT *
FROM table1
INNER JOIN table2
WHERE table1.id = table2.fk_id
```

FILTRE ORDRE

```
SELECT colonne1, colonne2
FROM table
ORDER BY colonne1
SELECT colonne1, colonne2, colonne3
FROM table
ORDER BY colonne1 DESC, colonne2
ASC
```



By **mr.freelancer**
cheatography.com/mr-freelancer/

Not published yet.
 Last updated 5th April, 2024.
 Page 1 of 3.

Sponsored by **CrosswordCheats.com**
 Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

FILTRE ORDRE (cont)

Par défaut les résultats sont classés par ordre ascendant

```
SELECT *
FROM utilisateur
ORDER BY nom
```

```
SELECT *
FROM utilisateur
ORDER BY nom, date_inscription DESC
```

Créer une table

```
CREATE TABLE nomDeLaTable (
  nomDuChamp TYPE options ,
  nomDuChamp TYPE options
);
typeS : VARCHAR(nbCaractère)
FLOAT
INTEGER
BOOLEAN
LONGTEXT
options:
DEFAULT {valeur}
NOT NULL
NULL
PRIMARY KEY
AUTO_INCREMENT
```

```
CREATE TABLE animal (
  id int(11) NOT NULL,
  nom varchar(255) NOT NULL,
  description longtext DEFAULT NULL,
  image varchar(255) NOT NULL,
  poids int(11) NOT NULL,
  dangereux tinyint(1) NOT NULL,
  famille_id int(11) NOT NULL
)
```

GROUP BY

```
SELECT colonne1, fonction(colonne2)
FROM table
GROUP BY colonne1
SELECT client, SUM(tarif)
FROM achat
GROUP BY client
```

La manière simple de comprendre le GROUP BY c'est tout simplement d'assimiler qu'il va éviter de présenter plusieurs fois les mêmes lignes. C'est une méthode pour éviter les doublons

Filtre Caractère

```
SELECT *
FROM client
WHERE ville LIKE 'N%'
LIKE '%a' : le caractère "%" est un caractère joker qui remplace tous les autres caractères.
```

Ainsi, ce modèle permet de rechercher toutes les chaînes de caractère qui se terminent par un "a".

LIKE 'a%' : ce modèle permet de rechercher toutes les lignes de "colonne" qui commencent par un "a".

LIKE '%a%' : ce modèle est utilisé pour rechercher tous les enregistrements qui utilisent le caractère "a".

LIKE 'pa%on' : ce modèle permet de rechercher les chaînes qui commencent par "pa" et qui se terminent par "on", comme "pantalon" ou "pardon".

LIKE 'a_c' : peu utilisé, le caractère "_" (underscore) peut être remplacé par n'importe quel caractère,

Modifier des données

Modifier une ligne :

```
UPDATE table
SET colonne_1 = 'valeur 1', colonne_2 = 'valeur 2', colonne_3 = 'valeur 3'
WHERE condition
```

Modifier toutes les lignes :

```
UPDATE table
SET colonne = 'valeur'
```

Modifier une ligne :

```
UPDATE client
SET rue = '49 Rue Ameline',
ville = 'Saint-Eustache-la-Forêt',
code_postal = '76210'
WHERE id = 2
```

Modifier toutes les lignes :

```
UPDATE client
SET pays = 'FRANCE'
```

LIMITE

```
SELECT *
FROM table
LIMIT 10
```

Cette requête permet de récupérer seulement les 10 premiers résultats d'une table.

```
SELECT *
FROM table
LIMIT 5, 10;
```

Cette requête retourne les enregistrements 6 à 15 d'une table. Le premier nombre est l'OFFSET tandis que le suivant est la limite.



Comparaison

= Égale
 <> Pas égale
 != Pas égale
 > Supérieur à
 < Inférieur à
 >= Supérieur ou égale à
 <= Inférieur ou égale à
 IN Liste de plusieurs valeurs possibles
 BETWEEN Valeur comprise dans un intervalle donné (utile pour les nombres ou dates)
 LIKE Recherche en spécifiant le début, milieu ou fin d'un mot.
 IS NULL Valeur est nulle
 IS NOT NULL Valeur n'est pas nulle
 NOT EXISTS contraire

Alias

```
SELECT column_name AS alias_name
FROM table_name;
SELECT column_name(s)
FROM table_name AS alias_name;
```

Alias

```
SELECT column_name AS alias_name
FROM table_name;
SELECT column_name(s)
FROM table_name AS alias_name;
```

Supprimer des données

Supprimer une ligne
 DELETE FROM table
 WHERE condition
 EX :
 DELETE FROM utilis_ateur
 WHERE id = 1

Supprimer toutes les données d'une table
 DELETE FROM nomTable

EX: DELETE FROM utilis_ateur

Sélectionné des données

```
SELECT nomColonne, nomColonne
FROM nomDeLaTable ;
```

```
SELECT 'id', nom FROM animal
```

AGREGA

AVG() pour calculer la moyenne d'un set de valeur. Permet de connaître le prix du panier moyen pour de chaque client
COUNT() pour compter le nombre de lignes concernées. Permet de savoir combien d'achats a été effectué par chaque client
MAX() pour récupérer la plus haute valeur. Pratique pour savoir l'achat le plus cher
MIN() pour récupérer la plus petite valeur. Utile par exemple pour connaître la date du premier achat d'un client
SUM() pour calculer la somme de plusieurs lignes. Permet par exemple de connaître le total de tous les achats d'un client

Créer des données

```
INSERT INTO nomDeLaTable
('colonne1', 'colonne2', 'colonne3')
VALUES
(valeur1, valeur2, valeur3,);
attention si la vateur de de type texte mettre des guillemets.
pour les booleans mettre en majuscule.
```

```
INSERT INTO `animal` (`id`, `nom`, `description`, `image`, `poids`, `dangereux`, `famille_id`)
VALUES
(36, 'Chien', 'Un animal domestique', 'chien.png', 10, 0, 13),
(39, 'Crocodile', 'Un animal très dangereux', 'croco.png', 500, 1, 14),
(40, 'Requin', 'Un animal marin très dangereux', 'requin.png', 800, 1, 15);
```

