

MySQL MEMO

- Version : 04/02/25 -

===== ACCES, PRIVILEGES

Connexion à mysql en tant que root : `mysql -u root -p`

Accès à MySQL par root :

-- for MySQL

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'root';
```

-- for MariaDB

```
ALTER USER 'root'@'localhost' IDENTIFIED VIA mysql_native_password USING PASSWORD('root');
```

!!!!!!! Marche pas avec mysql / MariaDB ??

Changement de mot de passe root :

```
sudo mysqladmin -u root -h localhost password 'password'
```

Création d'un utilisateur mysql :

```
CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';
```

Lister les utilisateurs d'une base de données (en ayant les privilèges nécessaires) :

```
SELECT user,host FROM mysql.user;
```

On accorde des privilèges à l'utilisateur :

```
GRANT ALL PRIVILEGES ON *.* TO 'user'@'localhost';
```

```
GRANT ALL PRIVILEGES ON my_database1.* to 'user'@'localhost';
```

On révoque les privilèges :

```
REVOKE ALL PRIVILEGES ON *.* FROM 'user'@'localhost';
```

Ou des privilèges particuliers :

```
REVOKE CREATE, DROP ON my_Database.* FROM 'user'@'localhost';
```

Paramètres de la commande GRANT :

. Toutes les bdd et tous leurs objets

database.* La seule bdd nommée "database" et tous ses objets

database.object La seule base de donnée nommée "database" et son objet nommé "object"

Vérifier les autorisations d'un utilisateur :

```
SHOW grants FOR 'user'@'localhost';
```

On valide les changements de manière effective : FLUSH PRIVILEGES

Re-assign host access permission to MySQL user

```
UPDATE mysql.user SET host = '10.0.0.%' WHERE host = 'localhost' AND user != 'root';
```

```
Mysql > FLUSH PRIVILEGES;
```

Changer un mot de passe avec mysqladmin :

```
mysqladmin -u root -p"mdp_root" password "nouveau_mdp"
```

Avec la commande mysql :

```
mysql -u root -p
```

On utilisera donc la commande "UPDATE". Par exemple pour changer le mot de passe de l'utilisateur root :

```
UPDATE mysql.user SET password=PASSWORD("nouveau_mdp") where User="root";
```

=====

Lister les bases de données :

SHOW DATABASES;

Renommer une base de donnée :

Faire un dump pour sauvegarder oldDB, create newDB, puis restauration de oldDB.sql dans newDB

Annuler l'écriture d'une ligne avec \c

exemple : ALTERR teygxsl \c

===== COMMANDES USUELLES :

ALTER	Modifier une bdd ou une table
BACKUP	Créer une sauvegarde d'une table
\c	Annuler l'entrée
CREATE	Créer une base de données
DELETE	Supprimer une ligne d'une table
DESCRIBE	Décrire les colonnes d'une table
DROP	Supprimer une bdd ou une table
EXIT	Quitter mysql
GRANT	Modifier les privilèges d'un utilisateur
HELP	Afficher l'aide
INSERT	Insérer des données
LOCK	Vérouiller une ou plusieurs tables
QUIT	EXIT
RENAME	Renommer une table
REVOKE	annuler des droits sur des bases / tables
SHOW	Enumérer les détails à propos d'un objet
SOURCE	Executer un fichier
STATUS	Afficher l'état courant
TRUNCATE	Vider une table
UNLOCK	Dévérouiller une ou plusieurs tables
UPDATE	Modifier une enregistrement existant
USE	Utiliser une bdd

===== CREATION / MODIFS

Créer une bdd :

CREATE DATABASE maDatabase CHARSET=utf8;

Création d'un table :

USE maDatabase;

CREATE TABLE maTable (auteur **VARCHAR**(128), titre **VARCHAR**(128), **type** **VARCHAR**(128), annee **CHAR**(4)) **ENGINE** MyISAM **CHARSET** utf8;

Renommer une colonne dans une table [MARCHE PAS AVEC MARIADB ???] :

ALTER TABLE maTable **RENAME COLUMN** oldName **TO** newName;

ALTER TABLE maTable **RENAME COLUMN** old_col_name **TO** new_col_name;

Fonctionne :

ALTER TABLE maTable **CHANGE** old_column_name new_column_name Type_Definitions;

Par exemple : alter table tableau-periodique change masse_atomimic masse_atomique float;

Afficher les autorisations d'un utilisateur :

SHOW GRANTS **FOR** 'user'@'localhost';

Delete / Drop :

DELETE FROM maTable **WHERE** prenom = 'Michel';

DROP TABLE maTable;

===== TYPES DE DONNEES

CHAR(n) -> 'Bonjour' occupe n octets, n<255
VARCHAR(n) -> VARCHAR(7) : 'Bonjour' occupe 7 octets VARCHAR(100) : 'Bonjour' occupe 10 (ou 11 ?) octets

BINARY(n) ou BYTE(n) -> occupe n octets, n<255, avec des données binaires
VARBINARY(n) -> occupe un nombre variable d'octets, n<65535 (comme VARCHAR)

BLOB : objets binaires dont la taille dépasse les 65536 octets.

TINYBLOB(n) -> n<255 Données binaires, sans jeu de caractères
BLOB(n) -> n<65535
MEDIUMBLOB(n) -> n<1.67e+7
LONGBLOB(n) -> n<4.29e+9

TEXT : les champs TEXT ne peuvent avoir de valeur par défaut, MySQL n'indexe que les n premiers caractères d'une colonne de type TEXT (len est précisé au moment de la création de l'index)
VARCHAR plus intéressant si on doit effectuer une recherche sur la totalité d'un champ, TEXT est valable lorsque l'on effectue une recherche à partir des premiers caractères d'un champ.

TINYTEXT(n) -> n<255, chaîne avec un jeu de caractères
TEXT(n) -> n<65535
MEDIUMTEXT(n) -> n<1.6e+7
LONGTEXT(n) -> n<4.29e+9

==== NUMERIQUE

TINYINT 255 valeurs (-128<n<128 en entiers signés)
SMALLINT 65535 valeurs
MEDIUMINT 1.67e+7 valeurs
INT 4.29e+9 valeurs
BIGINT 1.84e+19
FLOAT +-3.4e+38
DOUBLE, REAL +-1.8e+308

Si utilisation de UNSIGNED (non valable pour FLOAT, REAL, DOUBLE) > uniquement des valeurs positives.
ex : **CREATE TABLE** nomTable (nomChamp **INT UNSIGNED**);

===== DATES

```
select today2,pt100_tempe,today from tempe1 where DATE(today)=CURDATE();
```

> today2 et pt100_tempe représentent des champs de la table tempe1, de format DATE
> CURDATE renvoie la date courante, du type AAAA-MM-JJ ou AAAAMMJJ

Formats : DATE (**YYYY-MM-DD** format) - DATETIME (**YYYY-MM-DD HH:MM:SS** format)
TIMESTAMP (**HH:MM:SS** format) - TIMESTAMP (similaire DATETIME, diffère au niveau de la conversion locale>UTC)

Fonctions : CURDATE(), CURTIME(), NOW(), DATE(), EXTRACT(), DATE_ADD(), DATEDIFF()...

Exemple :

```
SELECT champ1,champ2 FROM table1  
WHERE DATE('champDate') > DATE_SUB(CURDATE(), INTERVAL 1 WEEK) LIMIT 10";
```

===== MODIFS

Ajouter une clé dans une table (la clé se nomme 'id'):

```
ALTER TABLE maTable ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY;
```

Modifier un indice existant (id) pour le transformer en clé primaire :

```
ALTER TABLE maTable2 ADD PRIMARY KEY (id);
```

Modifier maTable2 pour rajouter l'AUTO_INCREMENT :

```
ALTER TABLE maTable2 MODIFY id INT UNSIGNED NOT NULL AUTO_INCREMENT
```

Ajouter un enregistrement :

```
INSERT INTO maTable(champ1, champ2,...) VALUES(val1, val2,...);
```

Modifier la valeur d'un champ dans un enregistrement :

```
UPDATE maTable SET Auteur='Marcel Proust' WHERE Roman='Du côté de chez Swann';
```

Changer la position d'une colonne "colonne1" en la déplaçant après "colonne_0" :

```
ALTER TABLE maTable MODIFY colonne1 VARCHAR(16) AFTER colonne_0;
```

l'astuce ici : on utilise MODIFY (normalement pour changer le type, mais on garde le même -> VARCHAR(16)), pour ensuite la placer où l'on souhaite, en l'occurrence après "colonne_0".

```
SELECT auteur,titre FROM classiques WHERE auteur LIKE "Charle%" AND auteur NOT LIKE "%Darwin";
```

===== CLEFS ETRANGERES - clés étrangères - Foreign keys

```
ALTER TABLE appartenir ADD CONSTRAINT 'fk_idLivre_livre' FOREIGN KEY ('idLivre') REFERENCES livre(ISBN);
```

>> fk_idLivre_livre est alors une contrainte de clé étrangère - cette clé étrangère s'appelle 'idLivre', se trouve dans la table 'appartenir' qui fait le lien entre les tables 'livre' et 'genre' - Cette clé étrangère fait référence à la clé primaire de la table 'livre' dont le nom est 'ISBN'.

réponse phpMyAdmin 25/10/22 : #1015 impossible d'ajouter des contraintes d'adresses externes.

=====JOINTURES

Soient 2 tables : table1, table2

table1 a les champs : id1,nom,prenom,age,id2

table2 a les champs : id2,profession

```
SELECT nom,prenom
FROM table1
JOIN table2
ON table1.id2=table2.id2;
```

Clef primaire composée :

```
CREATE TABLE Personnes (
last_name VARCHAR(20) NOT NULL,
first_name VARCHAR(20) NOT NULL,
age int,
address VARCHAR(100),
PRIMARY KEY(last_name, first_name)
);
```

=====

SAUVEGARDES

=====

Sauvegarder 1 BASE :

```
mysqldump -u utilisateur -p mot_de_passe base > base.sql
```

Sauvegarder 1 table :

LOCK TABLES publications.classiques **READ**;
> permet de verrouiller la table à sauvegarder

```
mysqldump -u utilisateur -p publications classiques > classiques.sql
```

> à taper dans un autre Terminal

UNLOCK TABLES;
> pour supprimer le verrou

Sauvegarder toutes les tables :

```
mysqldump -u utilisateur -p --all-databases > tous_bdd.sql
```

=====

RESTAURATION

=====

Restaurer toutes les bases :

```
mysql -u utilisateur -p < toutes_bdd.sql
```

Restaurer toutes la base "publications" :

```
mysql -u utilisateur -p -D publications < publications.sql
```

Restaurer la table "classiques" dans le BDD "publications" :

```
mysql -u utilisateur -p -D publications < classiques.sql
```

Petits rappels sur les requêtes (Cours Greta – Jessie S.)

Se rappeler de l'ordre des clauses

SELECT

Ici on choisit les champs(ou nom de colonne ou "attributs").

Ce que l'on vous demande. Cela peut être un (ou des) champ simple (nom,...) mais aussi une fonction (COUNT,AVG,...) appliquée à un (ou des) champ(s).

FROM

Ici on trouve la table, ou les tables qui contiennent les champs que l'on vous demande dans le SELECT. Si c'est une table, pas de problème. Si les champs proviennent de plusieurs tables liées entre elles alors il faudra utiliser les jointures.

FROM table1 **JOIN** table2 **ON** id.exemple1=id.exemple2.

Ici les champs dont j'ai besoin sont dans les tables 1 et 2. J'ai donc besoin de ces 2 deux tables ou d'une seule qui les rassemble donc d'une jointure.

ON permet d'indiquer l'égalité des champs qui relient les deux tables.

Clé primaire/clé étrangère.

WHERE

WHERE est un filtre en entrée , effectué en 1^{er} par SQL.

Là, on cible les données plus précisément. Ce sont "les conditions" appliquées aux champs qui nous permettent d'accéder aux données demandées.

Attention il existe de nombreuses conditions : > ; < ; + ; = ; AND ; NOT ; IN ; LIKE...

Rappel : Pas de fonction d'agrégation dans un WHERE.

GROUP BY

Ici c'est le regroupement des résultats.

Toujours après WHERE et toujours avant HAVING.

Ce regroupement permet d'éviter de présenter des doublons.

On peut regrouper par taille, nom , sexe, salaire...

HAVING

Un peu comme WHERE => conditions.

C'est le filtre de sortie, et celui qui permet d'utiliser les fonctions d'agrégation. Très souvent utilise en même temps que GROUP BY.

ORDER BY

A la fin on trie les résultats. ASC par défaut. DESC possible

LIMIT

Permet de limiter le nombre de résultats de la requête.

LIMIT 5 ; Resultat limité à 5.

LIMIT 5,3 ; Resultat limité à 6,7,8. (5 est l'offset, 3 la limite).

OPERATEURS DE COMPARAISON

IN	Liste de plusieurs valeurs possibles
BETWEEN	Valeur comprise dans un intervalle donnée (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

IN > sélection sur des valeurs discrètes

SELECT colonne
FROM table
WHERE colonne **IN** (valeur1, valeur2, valeur3, ...)

« On souhaite toutes les informations sur les employés dont le salaire est soit 1720 euros soit 4500 euros »

SELECT * **FROM** employe
WHERE salaire **IN**(1720,4500);

BETWEEN > sélection sur un intervalle de données
Avec MySQL, les valeurs sont incluses.

SELECT colonne
FROM table
WHERE colonne **BETWEEN** 'valeur1' **AND** 'valeur2'

ex : *SELECT * FROM employe WHERE salaire BETWEEN 1870 AND 3000;*

ex : *SELECT nom FROM `employe` WHERE nom BETWEEN "A" AND "L"*
> le nom commence par A...L (ordre alphabétique)

LIKE > opérateur de comparaison d'égalité

LIKE ' D %' : tout ce qui commence par D...
LIKE '%D' : tout ce qui termine par D...
LIKE '%D%' : tout ce qui contient le caractère D...
LIKE ' Du%nd ' : tout ce qui commence par Du et finir par nd peut importe ce qu'il y a au milieu...
LIKE '1_6' : 1 underscore pour un seul caractère possible.

Ex : *SELECT nom,prenom FROM employe WHERE prenom LIKE 'A_____'*
> avec 5 underscore : recherche les noms de 6 caractères, commençant par A

OPERATEURS DE TRI

ORDER BY

SELECT nom,prenom **FROM** employe
ORDER BY nom DESC;

ordre par défaut : ASC – ordre inverse : DESC (décroissant)

CLAUSE LIMIT, CONCAT

```
SELECT *  
FROM table  
LIMIT 3
```

> on limite la requête aux 3 premiers dans l'ordre

```
SELECT *  
FROM table  
LIMIT 3,2
```

> avec offset : on limite la requête au 2 premiers résultats dans l'ordre en commençant par l'index 3 soit le 4eme dans la table.

On souhaite obtenir toutes les informations concernant les employés à partir du 4eme (index 3 : 4ième ligne dans la table, les indices commencent à 0) et en limitant à 2 le nombre de résultats.

CONCAT (concaténation)

```
SELECT CONCAT (nom, ' ', prenom) AS identite FROM table;
```

FONCTIONS D'AGGREGATION

COUNT (comptage)

```
SELECT COUNT(colonne) FROM table;
```

```
SELECT COUNT(departement) nombre_departement FROM collaborateur;
```

nombre_departement : nom attribué à la colonne du résultat.

MIN, MAX

```
SELECT MAX(salaire) salaire_max FROM collaborateur;
```

SUM

> somme

```
SELECT SUM(salaire) FROM collaborateur;
```

AVG

> moyenne

```
SELECT AVG(salaire) FROM collaborateur;
```

LES REGROUPEMENTS

```
SELECT sexe, AVG(salaire) AS moyenneSalaire  
FROM `collaborateur`  
GROUP BY sexe
```

> on regroupe les résultats par sexe.

Important: La clause GROUP BY s'effectue sur une colonne sélectionnée et on l'utilise avec une fonction d'agrégation.

CLAUSE HAVING

Similaire à Where, mais filtre en sortie.

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

```
SELECT departement, AVG(salaire)
FROM `collaborateur`
WHERE AVG(salaire)<2000
```

> génère une erreur car AVG (moyenne) n'est pas encore calculée au moment du filtrage.

```
SELECT departement, AVG(salaire)
FROM `collaborateur`
HAVING AVG(salaire)<2000
```

> OK

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
LIMIT
```

L'ordre est important !!!

=====

exemple de requête UPDATE :

```
update personne p join login l using(idPersonne) set p.login=l.uLogin,p.password=l.uPassword;
```

```
UPDATE employees SET isConfirmed=1 WHERE name LIKE 'da%' LIMIT 10
```

=====

Jointure interne

```
SELECT collaborateur.nom, collaborateur.prenom, departement.nom
FROM collaborateur
JOIN departement.nom
ON collaborateur.idDepartement=departement.idDepartement
```

```
SELECT c.nom, c.prenom, d.nom
FROM collaborateur c
JOIN departement d
USING (idDepartement)
```

FONCTIONS

DELIMITER //	DELIMITER //
CREATE FUNCTION testFonc (nombre INT)	CREATE FUNCTION pairImpair (nombre INT)
RETURNS varchar(30)	RETURNS varchar(30)
DETERMINISTIC	DETERMINISTIC
BEGIN	BEGIN
DECLARE tmpAge INT ;	
SELECT COUNT(idTable2) INTO tmpAge FROM table2 WHERE age2<nombre;	
IF (tmpAge>1)	IF ((nombre%2==0)
THEN RETURN "Supérieur à 1";	THEN RETURN "Nombre pair";
ELSE RETURN "Inférieur à 1";	ELSE RETURN "Nombre impair";
END IF ;	END IF ;
END //	END //
DELIMITER ;	DELIMITER ;

PROCEDURES

```
DELIMITER //  
CREATE PROCEDURE procDepA()  
BEGIN  
SELECT COUNT(departement)  
FROM departement  
WHERE departement_nom LIKE "A%";  
END //
```

```
DELIMITER //  
CREATE PROCEDURE procNbVille(OUT nb INT)  
BEGIN  
SELECT COUNT(ville_id) into nb FROM ville  
WHERE ville_departement = '29';  
END //
```

Declare, Set : variables locales
@ : variable globale / utilisateur

```
mysql> call procNbVille(@nombre);  
mysql> select @nombre;
```

TRIGGER

```
CREATE TRIGGER nomTrig  
BEFORE | UPDATE  
INSERT | DELETE | UPDATE ON nomTable  
FOR EACH ROW  
BEGIN  
---- Instructions ----  
END
```

```
DELIMITER //  
CREATE TRIGGER trgUpper  
BEFORE INSERT ON collaborateur  
FOR EACH ROW  
BEGIN  
SET NEW.nom = UPPER(NEW.nom);  
END //
```

IF CONDITION THEN... REPEAT... UNTIL

```
WHILE Condition DO  
– instructions –  
END WHILE
```

```
LOOP  
– Instructions --  
END LOOP
```

Manipulation des dates

Sélectionner tous les champs dont la date ("datetime", de type timestamp) a moins de 10 jours :

```
SELECT * FROM contact WHERE CURDATE() < TIMESTAMPADD(DAY, 10, datetime);
```

Lister le moteur utilisé dans chaque table d'une base de donnée :

```
SELECT TABLE_NAME, ENGINE  
FROM information_schema.TABLES  
WHERE TABLE_SCHEMA = 'nom_de_la_base';
```

Modifier le type de moteur d'une base :

exemple : on part d'une table 'laTable' avec un moteur MyISAM que l'on souhaite changer :

```
ALTER TABLE 'laTable' ENGINE=InnoDB
```