

Dernière mise-à-jour : 2020/01/30 03:27

## 105.3 - Gérer des données avec SQL (4/60)

### SQL

#### Chaînes de caractères

Les chaînes de caractères doivent être entourées de ' ou de “.

##### *Exemples*

'Linux est incroyable'

“Windows est .....”

“Je lui dit : ““Je t'aime”””

#### Nombres

Il existe 3 types de nombres :

##### Nombres Entiers

Une séquence de chiffres **sans espaces**

##### *Exemple*

999256

0

## Nombres Décimaux

Utilisent le **point** comme séparateur.

*Exemple*

120.54

5566.8956e+12

## Nombres Négatifs

Sont précédés par le signe **-**.

*Exemple*

-458

-147.36

## Valeurs NULL

Une chaîne **sans données**.

<note warning> N'est pas la même chose qu'une chaîne **vide** ou un **0** dans le cas d'un nombre. </note>

## Noms de Fichiers

Les noms de bases, tables et colonnes :

- peuvent contenir jusqu'à 64 caractères
- peuvent commencer par un chiffre
- ne peuvent pas contenir **que** de chiffres
- ne peuvent pas contenir un /, une \ ou un **point**

Les alias :

- peuvent contenir jusqu'à 255 caractères
- peuvent contenir un /, une \ ou un **point**

## Variables Utilisateurs

Les variables n'ont pas besoin d'être **initialisées**. Elles sont automatiquement créées avec la valeur **NULL**.

## Commentaires

Les commentaires **d'une ligne** sont précédés par le caractère # ou --.

Les commentaires sous plusieurs lignes sont entourés comme suit :

```
/*
Ceci est
un commentaire
sur
plusieurs lignes
*/
```

## Commandes

### SELECT

Obtenir un ensemble de données à partir d'une ou de plusieurs tables.

#### Syntaxe

```
SELECT [table.][colonne]|expression [AS nom][,[table.][colonne]
FROM nom-table [, nom_table ...]
[WHERE condition [AND|OR condition ...]
[GROUP BY [table.][colonne],[table].[colonne] ...]
[ORDER BY [table.][colonne][Description],[table].[colonne] ...]
```

#### Exemples

```
SELECT nom, prenom FROM familles ORDER BY nom
```

```
SELECT nom, prenom FROM familles GROUP BY ville
```

### UPDATE

Mettre à jour des données dans une table.

#### Syntaxe

```
UPDATE [LOW_PRIORITY][IGNORE] nom_table
SET colonne = expression
WHERE expression
ORDER BY expression
LIMIT valeur
```

Directive	Description
LOW_PRIORITY	L'opération se fera quand la table n'est pas utilisée par un client
IGNORE	La mise à jour continue malgré des problèmes éventuels rencontrés. Les enregistrements posant problème ne seront PAS mis à jour

### Exemples

```
UPDATE familles SET Adresse2='*****', nb_enfants=2;
```

```
UPDATE familles SET nb_enfants=8 ORDER BY Nom LIMIT 3;
```

## DELETE FROM

Supprimer des enregistrements d'une table.

### Syntaxe

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM nom_table
      [WHERE where_definition]
      [ORDER BY ...]
      [LIMIT row_count]
```

### Exemple

```
DELETE FROM familles WHERE Nom = Durant;
```

## DROP TABLE

### Syntaxe

```
DROP DATABASE [IF EXISTS] db_name
```

Supprimer une table d'une base de données.

*Exemple*

```
DROP TABLE Test;
```

## INSERT

### Syntaxe

```
INSERT [LOW_PRIORITY] | [DELAYED] [IGNORE]
[INTO] nom_table [(nom_colonne ...)]
VALUES ((expression | DEFAULT) ...), (...)
```

ou

```
INSERT [LOW_PRIORITY] | [DELAYED] [IGNORE]
[INTO] nom_table [(nom_colonne ...)]
SELECT ...
```

Directive	Description
LOW_PRIORITY	L'opération se fera quand la table n'est pas utilisée par un client
DELAYED	Permet à un client d'utiliser la table sans attendre la fin de l'opération
IGNORE	La mise à jour continue malgré des problèmes éventuels rencontrés. Les enregistrements posant problème ne seront PAS mis à jour

## INSERT VALUES

Insérer des valeurs dans une table.

*Exemple*

```
INSERT INTO familles (Nom, Prenom) VALUES ('Smith', 'John');
```

## INSERT SELECT

Insérer des valeurs en provenance d'une autre table dans la table cible.

*Exemple*

```
INSERT INTO familles (Prenom) SELECT enfants.prenom FROM enfants;
```

## ALTER

Ajouter, supprimer ou modifier une colonne, index ou nom de table.

### ALTER ADD

Commande	Description
ADD [COLUMN]	Ajouter un champ
ADD INDEX	Ajouter un index
ADD PRIMARY KEY	Ajouter une clef primaire
ADD UNIQUE	Ajouter un index unique
ADD FULLTEXT	Ajouter une recherche texte entier

*Exemples*

```
ALTER TABLE familles ADD Sports VARCHAR(50) NOT NULL;
```

```
ALTER TABLE familles ADD INDEX (Sports);
```

### ALTER CHANGE

Permet de modifier le nom ou le type d'une colonne.

*Exemple*

```
ALTER TABLE familles CHANGE PrenomPere Prenom_Pere VARCHAR(45) NOT NULL;
```

## ALTER DROP

Commande	Description
DROP [COLUMN]	Supprimer un champ
DROP INDEX	Supprimer un index
DROP PRIMARY KEY	Supprimer une clef primaire

*Exemple*

```
ALTER TABLE familles DROP Sports;
```

## ALTER KEYS

Permet de désactiver la clef primaire momentanément.

*Exemples*

```
ALTER TABLE familles DISABLE KEYS;
```

```
ALTER TABLE familles ENABLE KEYS;
```

## ALTER RENAME

Permet de renommer une table.

*Exemple*

```
ALTER TABLE enfants RENAME familles_enfants;
```

## ALTER ORDER BY

Permet de reclasser physiquement une table.

### Exemple

```
ALTER TABLE familles ORDER BY Prenom_Pere;
```

### MATCH

Permet la recherche d'un mot, d'une phrase ou d'une expression sur un **texte entier**

### Exemple

```
SELECT * FROM familles WHERE MATCH (Commentaire) AGAINST ('fleuve');
```

## Opérateurs

### Mathématiques

Nom	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
()	Le calcul entre parenthèses est effectué en priorité

### Logiques

Nom	Description
!	NON
	OU
&&	ET
XOR	XOR exclusif

## Comparaison

Nom	Description
=	Egal
!= ou <>	Non égal
<	Inférieur
< =	Inférieur ou égal
>	Supérieur
> =	Supérieur ou égal
< = >	Egal gérant la nullité
IS NULL	Nullité
IS NOT NULL	Non nullité

## Fonctions

### Mathématiques

Nom	Description
ABS(nbr)	Valeur absolue de nbr
SIGN(nbr)	-1 0 ou 1 en fonction du signe de nbr
MOD(nbr1, nbr2)	Indique le reste de la division de nbr1 par nbr2
FLOOR(nbr)	Indique le plus grande nombre entier inférieur à nbr
CEILING(nbr)	Indique la plus petite valeur entière supérieure à nbr
ROUND(nbr)	Indique l'arrondi entier le plus proche de nbr
ROUND(nbr,D)	Indique l'arrondi à D décimales plus plus proche de nbr
EXP(nbr)	Indique l'exponentielle de nbr
POW(nbr,nbr2)	indique nbr à la puissance nbr2
SQRT(nbr)	Indique la racine carrée de nbr
PI()	Indique la valeur de PI
COS(nbr)	Indique le cosinus de nbr
ACOS(nbr)	Indique l'arc cosinus de nbr

Nom	Description
SIN(nbr)	Indique le sinus de nbr
ASIN(nbr)	Indique l'arc sinus de nbr
TAN(nbr)	Indique le tangent de nbr
ATAN(nbr)	Indique l'arc tangent de nbr
COT(nbr)	Indique la cotangente de nbr
RAND()	Indique une valeur en virgule flottante entre 0 et 1.0
RAND(nbr)	Indique une valeur en virgule flottante entre 0 et 1.0. La valeur de nbr indique la limite haute
LEAST(nbr1, nbr2 ...)	Indique le plus petit des nombres nbr1 à nbrX
GREATEST(nbr1, nbr2 ...)	Indique le plus grand des nombres nbr1 à nbrX
DEGREES(nbr)	Convertit nbr de gradiants vers degrés
RADIANS(nbr)	Convertit nbr de degrés vers gradiants
TRUNCATE(nbr,D)	Indique nbr tronqué à D décimales

## Chaînes

Nom	Description
LIKE	Effectue une comparaison en fonction d'un motif
NOT LIKE	L'inverse de LIKE
_	Remplace un caractère dans le motif d'une chaîne
%	Remplace un ou plusieurs caractère(s) dans le motif d'une chaîne
\	Caractère d'échappement
BINARY	Rend la comparaison avec un motif sensible à la casse
strcmp(chaîne1, chaîne2)	Compare deux chaînes. Retourne -1 si chaîne 1 < chaîne 2. Retourne 0 si chaîne 1 = chaîne 2. Retourne 1 si chaîne 1 > chaîne 2.
MATCH	Utilisé pour des recherches de texte intégral
UPPER('chaîne')	Transforme les minuscules en majuscules
LOWER('chaîne')	Transforme les majuscules en minuscules

## Dates

Nom	Description
NOW()	Retourne la date au format 'AAAA-MM-JJ HH:MM:SS'
DATE_FORMAT(date,format)	Retourne la date selon le format spécifié
DAYOFWEEK(date)	Retourne un chiffre qui représente le jour de la semaine ( 1 pour dimanche, 7 pour samedi )
WEEKDAY(date)	Retourne un chiffre qui représente le jour de la semaine ( 7 pour dimanche, 0 pour lundi )
DAYOFMONTH(date)	Retourne un chiffre de 1 à 31
DAYOFYEAR(date)	Retourne un chiffre de 1 à 366
MONTH(date)	Retourne un chiffre de 1 à 12
DAYNAME(date)	Retourne le nom du jour ( lundi, mardi ... )
MONTHNAME(date)	Retourne le nom du mois
QUARTER(date)	Retourne un chiffre de 1 à 4 ( 1 = premier trimestre de l'année )
WEEK(date [,depart])	Retourne une valeur de 1 à 52. La valeur de <i>depart</i> peut être 0 ou 1. Dans le cas de 0, le dimanche est considéré comme le premier jour de la semaine. Dans le cas de 1, c'est le lundi.
YEAR(date)	Retourne un chiffre de 1000 à 9999
HOUR(date)	Retourne l'heure
MINUTE(date)	Retourne les minutes
SECOND(date)	Retourne les secondes
TO_DAYS(date)	Retourne le nombre de jours écoulés depuis le début de l'an 0
FROM_DAYS(date)	Retourne la date à partir d'un nombre de jours écoulés depuis le début de l'an 0
CURDATE()	Retourne la date courante au format AAAA-MM-JJ
CURRENT_DATE()	Retourne la date courante au format AAAA-MM-JJ
CURTIME()	Retourne la date courante au format HH:MM:SS
CURRENT_TIME()	Retourne la date courante au format HH:MM:SS
UNIX_TIMESTAMP([date])	Retourne le nombre de secondes depuis la date 1970-01-01 00:00:00
FROM_UNIXTIME(unix_timestamp[,format])	Retourne la date en fonction d'un nombre de secondes depuis la date 1970-01-01 00:00:00

### Motifs

Nom	Description	Exemple
%M	Le mois	janvier
%W	Le jour de la semaine	lundi

Nom	Description	Exemple
%D	La date du mois	1st
%Y	L'année	2009
%y	L'année	09
%a	L'abréviation du jour	lun
%d	Le jour du mois	01
%m	Le numéro du mois	01
%b	L'abréviation du mois	lun
%j	Le jour de l'année	031
%H	L'heure	00
%h	L'heure	12
%i	Les minutes	05
%r	L'heure au format américain	11:01:00 PM
%T	L'heure au format 24 heures	23:01:00
%S	Les secondes	06
%p	AM ou PM	PM
%w	Le numéro du jour de la semaine	0 (=dimanche)
%U	Le numéro de la semaine avec le début de la semaine étant un dimanche	03
%u	Le numéro de la semaine avec le début de la semaine étant un lundi	02

Par exemple :

```
mysql> select DATE_FORMAT(now(), '%W %d %M %Y');
+-----+
| DATE_FORMAT(now(), '%W %d %M %Y') |
+-----+
| Friday 19 October 2012                |
+-----+
1 row in set (0.00 sec)

mysql>
```

## Contrôle

Nom	Description
IF(exp1, exp2, exp3)	Si exp1 est vrai, exp2 est retournée sinon exp3 est retournée
IFNULL(exp1,exp2)	Si exp1 est NULL, exp2 est retournée sinon exp1 est retournée
NULLIF(exp1,exp2)	Si exp1 = exp2, NULL est retournée sinon exp1 est retournée
CASE value WHEN comp1 THEN res1 [WHEN comp2 THEN res2][ELSE elseres] END	La fonction compare <i>value</i> à <i>comp1</i> ( <i>comp2</i> ... ). Si une égalité est trouvée, <i>res1</i> ( <i>res2</i> ... ) est retournée. Si aucune égalité n'est trouvée, <i>elseres</i> est retournée

## Agrégation

Nom	Description
AVG(colonne)	Moyenne de la colonne
COUNT(items)	Le nombre de valeurs non nulles de la colonne
MIN(colonne)	Valeur minimum de la colonne
MAX(colonne)	Valeur maximum de la colonne
STD(colonne)	Écart type des valeurs de la colonne
SUM(colonne)	Somme des valeurs de la colonne
BIT_OR(colonne)	Ou logique effectué sur les valeurs de la colonne
BIT_AND(colonne)	ET logique effectué sur les valeurs de la colonne

## Autres

Nom	Description
CAST(expression) CONVERT(expression)	Convertit l'expression vers le type demandé - <i>BINARY, DATE, DATETIME, SIGNED, TIME, UNSIGNED</i>
LAST_INSERT_ID()	Retourne la valeur d'une colonne <i>AUTO_INCREMENT</i> lors du dernier <i>insertion</i>
VERSION()	La version du serveur MySQL
CONNECTION_ID()	L'identifiant du thread de la connexion courante
DATABASE()	La base de données courante
USER()	L'utilisateur courant
PASSWORD(chaîne)	Encrypte un mot de passe
ENCRYPT(chaîne, [,force])	Encrypte un mot de passe avec la fonction <i>crypt</i> d'Unix

Nom	Description
ENCODE(chaîne,mdp)	Encode la chaîne avec le mot de passe <i>mdp</i>
DECODE(chaîne,mdp)	Décode la chaîne avec le mot de passe <i>mdp</i>
MD5(chaîne)	Retourne une chaîne encodée à la norme MD5
SHA1()	Retourne une chaîne encodée à la norme SHA1

## Types de Champs

### Nombres entiers

Type	Intervalle	Taille en octets	Description
TINYINT[(M)]	-127 à 128	1	Entiers très courts
TINYINT[(M)] UNSIGNED	0 à 255	1	Entiers très courts
SMALLINT[(M)]	-32 768 à 32 767	2	Entiers courts
SMALLINT[(M)] UNSIGNED	0 à 65 535	2	Entiers courts
MEDIUMINT[(M)]	- 8 388 608 à 8 388 607	3	Entiers de taille moyenne
MEDIUMINT[(M)] UNSIGNED	0 16 777 215	3	Entiers de taille moyenne
INT[(M)]	- $2^{31}$ à $2^{31} - 1$	4	Entiers
INT[(M)] UNSIGNED	0 à $2^{32} - 1$	4	Entiers
INTEGER[(M)]	-	-	Synonyme de INT
BIGINT[(M)]	- $2^{63}$ à $2^{63} - 1$	8	Entiers larges
BIGINT[(M)] UNSIGNED	0 à $2^{64} - 1$	8	Entiers larges

### Nombres à virgule flottante

Type	Intervalle	Taille en octets	Description
FLOAT(precision)	Varie selon <i>precision</i>	Varie	< =24 pour un nombre en simple précision ou >24 et < =53 pour un nombre en double précision
FLOAT[(M,D)]	+ ou - 1.175494351E-38 à + ou - 3.402823466E+38	4	Simple précision

Type	Intervalle	Taille en octets	Description
DOUBLE[(M,D)]	+ ou - 1.7976931348623157E+308 à + ou - 2.2250738585072014E-308	8	Double précision
DOUBLEPRECISION[(M,D)]	-	-	Synonyme de DOUBLE[(M,D)]
REAL[(M,D)]	-	-	Synonyme de DOUBLE[(M,D)]
DECIMAL[(M[,D])]	Varie	M + 2	Nombre à virgule flottante
NUMERIC	-	-	Synonyme de DECIMAL

## Dates et Heures

Type	Intervalle
DATE	1000-01-01 à 9999-12-31
TIME	-838:59:59 à 838:59:59
DATETIME	1000-01-01 00:00:00 à 9999-12-31 23:59:59
TIMESTAMP[(M)]	Voir tableau ci-dessous
YEAR[(2)]	70 à 69 (1970 à 2069)
YEAR[(4)]	1901 à 2155

## Types de données TIMESTAMP

Type	Affichage
TIMESTAMP	YYYYMMDDHHMMSS
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

## Chaînes

Type	Longeur	Description
CHAR(M)	1 à 255	Chaîne de longueur fixe où M varie entre 1 et 255
VARCHAR(M)	1 à 255	Chaîne de longueur variable où M varie entre 1 et 255

## TEXT et BLOB

- **TEXT**,
  - TEXT respecte la casse des données,
- **BLOB** (Binary Large OBject),
  - BLOB permet de stocker toute information binaire telle une image ou un son.

Type	Longueur Maximale en caractères	Description
TINYBLOB	255	Objet binaire court
TINYTEXT	255	Texte court
BLOB	65 535	Objet binaire de taille normale
TEXT	65 535	Texte de taille normale
MEDIUM BLOB	16 777 215	Objet binaire de taille moyenne
MEDIUM TEXT	16 777 215	Texte de taille moyenne
LONG BLOB	4 294 967 295	Objet binaire de grande taille
LONG TEXT	4 294 967 295	Texte de grande taille

## ENUM et SET

- **ENUM**,
  - permet de prendre une valeur ou NULL parmi une liste de valeurs prédéfinies à la création de la colonne,
- **SET**
  - permet de prendre un maximum de 64 valeurs ou NULL parmi une liste de valeurs prédéfinies à la création de la colonne,

Type	Maximum des valeurs dans l'ensemble	Description
ENUM('valeur1','valeur2',...)	65 535	Les valeurs de ce type ne peuvent contenir qu'une seule des valeurs de la liste ou NULL

Type	Maximum des valeurs dans l'ensemble	Description
SET('valeur1','valeur2',...)	64	Les valeurs de ce type ne peuvent contenir un ensemble des valeurs de la liste ou NULL

## Types de Moteurs de Stockage

Le type de moteur est spécifié lors de la création de la table. Le moteur d'une table existant peut être modifié avec la commande **ALTER TABLE**. Les différents types **principaux** sont résumés ci-après :

Type de Moteur	Description
InnoDB	Le format par défaut. Supportent les transactions sécurisées avec <b>COMMIT</b> et <b>ROLLBACK</b> . Permet de verrouiller des enregistrements un-à-un au lieu de la table entière.
MyISAM	Données stockées dans un fichier .MYD. Index stockés dans un fichier .MYI. Structure de la table stockée dans un fichier .frm
MEMORY	Anciennement connues sous le nom HEAP. Les données sont stockées en mémoire tandis que la structure de la table est stockée sur disque dans un fichier .frm. Ne supportent pas les champs TEXT, BLOB. Ne supportent pas l'attribut AUTO_INCREMENT. Utilisé pour augmenter la vitesse de traitement d'une requête.
ARCHIVE	Ne supportent que des requêtes de lecture et d'insertion. Utilisé pour stocker des données des journaux d'application. Structure stockée dans un fichier .frm. Données stockées dans un fichier .ARZ. Métadonnées sont stockées dans un fichier .ARM
CSV	Données sont stockées dans un fichier texte au format CSV. Utilisé pour échanger des données avec d'autres applications. Structure stockée dans un fichier .frm. Données stockées dans un fichier .CSV.
FEDERATED	Ne stocke pas de données. Extrait des données de tables en provenance de bases sur des serveurs distants.

Bien que possible techniquement, il n'est pas préférable d'utiliser des types de moteurs différents dans la même base de données car ceci complique la tâche d'optimisation, augmente le nombre de buffers et de caches et complexifie le travail de l'administrateur.

Le choix d'un moteur de stockage se fait en fonction de l'adéquation entre les besoins et les caractéristiques suivants :

Moteur	Verrous	Transactionnel de type <b>ACID</b>	MVCC	Clef Etrangère	Données sur Disque	Sauvegarde à Chaud	COMMIT	ROLLBACK	Crash Recovery
InnoDB	Enregistrement	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui
MyISAM	Table	Non	Non	Non	Oui	Non	Non	Non	Non
Memory	Table	Non	Non	Non	Non	Non	Non	Non	Non
Archive	Enregistrement	Non	Non	Non	Oui	Non	Non	Non	Non

Moteur	Verrous	Transactionnel de type <b>ACID</b>	MVCC	Clef Etrangère	Données sur Disque	Sauvegarde à Chaud	COMMIT	ROLLBACK	Crash Recovery
CSV	Table	Non	Non	Non	Oui	Non	Non	Non	Non
Federated	Table	Non	Non	Non	Oui	Non	Non	Non	Non
NDB Cluster	Enregistrement	Oui	Non	Oui	Oui	Oui	Oui	Oui	Oui

## Jointures

Les jointures permettent de créer un lien entre deux tables. Le champ utilisé pour la jointure dans la première table est souvent la **clef primaire**. Le champs utilisé dans la deuxième table est appelé le **clef étrangère**. La clef primaire et la clef étrangère doivent être du même **type**.

Dans le cas où on utilise un champs autre que la clef primaire pour assurer la jointure, cet autre champs doit être **indexé**.

### FULL JOIN

Ce type de jointure permet de renvoyer uniquement les enregistrements des **deux** tables 1 et 2 ayant une correspondance :

```
mysql> SELECT table1.champ1, table2.champ1 FROM table1, table2 WHERE table1.clefprimaire = table2.clefetrangere;
```

ou

```
mysql> SELECT table1.champ1, table2.champ1 FROM table1 JOIN table2 WHERE table1.clefprimaire = table2.clefetrangere;
```

### LEFT JOIN

Ce type de jointure permet de renvoyer les enregistrements de la première table qui ont une correspondance dans la deuxième table. La syntaxe est la suivante :

```
mysql> SELECT table1.champ1, table2.champ1 FROM table1 LEFT JOIN table2 ON table1.clefprimaire =
```

```
table2.clefetrangere;
```

Le résultat de cette requête est l'affichage de **tous** les enregistrements de la **table 1** avec ou sans les enregistrements correspondants de la **table 2**. Dans le cas où la table 2 ne contient pas d'enregistrement correspondant à un enregistrement de la table 1, la valeur renournée est **NULL**. Cette information nous permet de rechercher uniquement les enregistrements dans la table1 n'ayant **pas** d'enregistrements dans la table 2 :

```
mysql> SELECT table1.champ1, table2.champ1 FROM table1 LEFT JOIN table2 ON table1.clefprimaire =  
table2.clefetrangere WHERE table2.clefetrangere IS NULL;
```

## RIGHT JOIN

Ce type de jointure est l'inverse d'un LEFT JOIN.

## LAB #1 - Le Langage SQL

Créez maintenant la base de données **CarnetAdresses** :

```
mysql> CREATE DATABASE `CarnetAdresses` ;  
Query OK, 1 row affected (0.00 sec)  
  
mysql>
```

Créez ensuite deux tables **familles** et **enfants** dans la base **CarnetAdresses** :

```
mysql> USE CarnetAdresses ;  
Database changed  
mysql>  
  
mysql> CREATE TABLE familles (CodeFamille BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY, Nom VARCHAR(40),  
-> PrenomPere VARCHAR(40) , PrenomMere VARCHAR(40) , Adresse1 VARCHAR(40) , Adresse2 VARCHAR(40),
```

```
-> CodePostal VARCHAR(5), Ville VARCHAR(40), ProfPere VARCHAR(40), ProfMere VARCHAR(40));  
Query OK, 0 rows affected (0.03 sec)
```

```
mysql>
```

Pour plus de facilité, copiez simplement la requête et collez-la dans votre terminal :

[familles](#)

```
CREATE TABLE familles (CodeFamille BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY, Nom VARCHAR(40),\\  
PrenomPere VARCHAR(40) , PrenomMere VARCHAR(40) , Adresse1 VARCHAR(40) , Adresse2 VARCHAR(40),\\  
CodePostal VARCHAR(5), Ville VARCHAR(40), ProfPere VARCHAR(40), ProfMere VARCHAR(40));
```

```
mysql> CREATE TABLE Enfants ( CodeFamille BIGINT NOT NULL, Prenom VARCHAR(40) NOT NULL,  
-> Sexe VARCHAR(1) NOT NULL, DateNaissance DATE NOT NULL, PRIMARY KEY (CodeFamille));  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql>
```

Pour plus de facilité, copiez simplement la requête et collez-la dans votre terminal :

[enfants](#)

```
CREATE TABLE Enfants ( CodeFamille BIGINT NOT NULL, Prenom VARCHAR(40) NOT NULL,\\  
Sexe VARCHAR(1) NOT NULL, DateNaissance DATE NOT NULL, PRIMARY KEY (CodeFamille));
```

Utilisez ensuite la commande SHOW pour visualiser le résultat de chaque instruction, par exemple :

```
mysql> SHOW DATABASES;  
+-----+  
| Database |
```

```
+-----+  
| information_schema |  
| CarnetAdresses |  
| mysql |  
| test |  
+-----+  
4 rows in set (0.00 sec)
```

```
mysql>
```

```
mysql> SHOW TABLES FROM CarnetAdresses;  
+-----+  
| Tables_in_CarnetAdresses |  
+-----+  
| Enfants |  
| familles |  
+-----+  
2 rows in set (0.00 sec)
```

```
mysql>
```

```
mysql> SHOW COLUMNS FROM familles FROM CarnetAdresses;  
+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| CodeFamille | bigint(20) | NO | PRI | NULL | auto_increment |  
| Nom | varchar(40) | YES | | NULL | |  
| PrenomPere | varchar(40) | YES | | NULL | |  
| PrenomMere | varchar(40) | YES | | NULL | |  
| Adresse1 | varchar(40) | YES | | NULL | |  
| Adresse2 | varchar(40) | YES | | NULL | |  
| CodePostal | varchar(5) | YES | | NULL | |  
| Ville | varchar(40) | YES | | NULL | |  
| ProfPere | varchar(40) | YES | | NULL | |
```

```
| ProfMere | varchar(40) | YES |      | NULL   |  
+-----+-----+-----+-----+-----+  
10 rows in set (0.00 sec)
```

mysql>

```
mysql> SHOW INDEX FROM familles;  
+-----+-----+-----+-----+-----+-----+-----+  
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed |  
| Null | Index_type | Comment |  
+-----+-----+-----+-----+-----+-----+-----+  
| familles |      0 | PRIMARY |          1 | CodeFamille | A          |      0 | NULL | NULL |  
| BTREE   |      |          |  
+-----+-----+-----+-----+-----+-----+-----+  
+-----+-----+  
1 row in set (0.00 sec)
```

mysql>

```
mysql> SHOW TABLE STATUS FROM CarnetAdresses;  
+-----+-----+-----+-----+-----+-----+-----+-----+  
| Name    | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length |  
| Data_free | Auto_increment | Create_time |          | Update_time | Check_time | Collation | Checksum |  
| Create_options | Comment |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
+-----+-----+-----+-----+-----+-----+-----+-----+  
+-----+-----+  
| Enfants | InnoDB |      10 | Compact    |    0 |          0 |      16384 |          0 |          0 |  
| 9437184 |          NULL | 2014-01-27 10:46:10 | NULL |          | NULL | latin1_swedish_ci |          | NULL |  
|          |  
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
| familles | InnoDB |      10 | Compact    |    0 |          0 |          16384 |          0 |          0 |
| 9437184 |           1 | 2014-01-27 10:45:39 | NULL    | NULL      | latin1_swedish_ci | NULL      |
|          |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
```

Créez les enregistrements dans la table **familles** :

data

```
INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Durant', 'Jacques', 'Jane', '23 rue Dutor','','92200', 'Neuilly', 'Plombier','');
INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Tortua', 'Benoît', 'Carole', '7 rue Verget','','75005', 'Paris', 'Cadre', 'Cadre');
INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Portier', 'Pierre', 'Elisabeth', '5 rue Toulet','','92200', 'Neuilly', 'Dentiste','');
INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Renault', 'Damien', 'Anne', '2 rue Ragon','','75007', 'Paris', 'Comptable',
'Enseignante'); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal ,
Ville , ProfPere , ProfMere) VALUES ('Darduet', 'Jean','','','','','','',''); INSERT INTO familles (Nom ,
PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES ('Rodier',
'Gérard', 'Aurélie', '6 rue Agien', '77000', 'Fontainebleau', 'Professeur','');
INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES ('Tarte',
'Alla', 'Crème', '1 allée Durond','','92200', 'Neuilly', 'Cuisinier', 'Cuisinière'); INSERT INTO familles
(Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES
('Cohen', 'David', 'Sarah', '7 Av d'Eylau','','75016', 'Paris', 'PDG', 'Assistante Direction'); INSERT
INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Dupont2', 'Bruno', 'Odile', '12 rue Sébastien','','75008', 'PARIS', 'Electricien',
'Comptable'); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville
, ProfPere , ProfMere) VALUES ('Durand2', 'Jacques', 'Jane', '23 rue Dutor','','92200', 'Neuilly',
```

```
'Plombier',''); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES ('Darmon2', 'Bruno', 'Béatrice', '2 rue Nicolo','','13008', 'Marseille', 'Cadre', 'Peintre'); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES ('Darduet2', 'Jean', '','','','','',''); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES ('Tarte2', 'Alla', 'Crème', '1 allée Durond','','92200', 'Neuilly', 'Cuisinier', 'Cuisinière'); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES ('Cohen2', 'David', 'Sarah', '7 Av d'Eylau','','75016', 'Paris', 'PDG', 'Assistante Direction');
```

Saisissez maintenant chacune des commandes suivantes et expliquez le résultat obtenu. Les instructions sont numérotées pour faciliter vos prises de notes.

1	ALTER TABLE Enfants ADD INDEX (Prenom);
2	ALTER TABLE Enfants DROP INDEX Prenom;
3	ALTER TABLE Enfants ADD TelPort VARCHAR(50) NOT NULL;
4	ALTER TABLE Enfants CHANGE TelPort TelPort VARCHAR( 22 ) NOT NULL;
5	ALTER TABLE Enfants ADD Test VARCHAR( 40 ) NOT NULL;
6	ALTER TABLE Enfants DROP Test;
7	SELECT * FROM familles;
8	SELECT * FROM familles LIMIT 8,12;

<note tip> **LIMIT** - Avec un argument, la valeur spécifie le nombre de lignes à retourner depuis le début du jeu de résultat. Si deux arguments sont donnés, le premier indique le décalage du premier enregistrement à retourner, le second donne le nombre maximum d'enregistrement à retourner. Le décalage du premier enregistrement est 0 (pas 1). </note>

9	UPDATE familles SET Adresse1 = '120 rue de Vaugirard', CodePostal = '75015', Ville = 'PARIS' WHERE Nom = 'DARDUET' AND PrenomPere = 'Jean' LIMIT 1;
10	DELETE FROM familles WHERE Nom = 'DURANT' AND PrenomPere = 'Jacques';
11	DELETE FROM familles WHERE Ville = 'Neuilly';
12	SELECT * FROM familles LIMIT 8,12;
13	SELECT nom, PrenomPere FROM familles;

```

14| SELECT Nom AS 'Nom de famille', PrenomPere AS 'Prenom du Père' FROM familles;
15| INSERT INTO Enfants (Prenom, Sexe, CodeFamille) VALUES ('Georges', 'M', '14');
16| SELECT familles.nom, familles.PrenomPere FROM familles, Enfants;

```

<note> Créez la colonne **familles.nb\_enfants**. </note>

```

17| UPDATE familles SET nb_enfants = '2' WHERE CodeFamille = '2';
18| UPDATE familles SET nb_enfants = '1' WHERE CodeFamille = '4';
19| UPDATE familles SET nb_enfants = '3' WHERE CodeFamille = '5';
20| UPDATE familles SET nb_enfants = '2' WHERE CodeFamille = '6';
21| UPDATE familles SET nb_enfants = '4' WHERE CodeFamille = '8';
22| UPDATE familles SET nb_enfants = '5' WHERE CodeFamille = '9';
23| UPDATE familles SET nb_enfants = '3' WHERE CodeFamille = '11';
24| UPDATE familles SET nb_enfants = '1' WHERE CodeFamille = '12';
25| UPDATE familles SET nb_enfants = '5' WHERE CodeFamille = '14';
26| SELECT SUM(nb_enfants) FROM familles;
27| SELECT MIN(nb_enfants) AS 'Nb enfants minimum', MAX(nb_enfants) AS 'Nb enfants maximum', AVG(nb_enfants) AS 'Nb enfants moyen' FROM familles;
28| SELECT nom FROM familles WHERE ville = 'Paris';
29| SELECT nom FROM familles WHERE ville = 'Paris' OR ville = 'Neuilly';
30| ALTER TABLE Enfants DROP PRIMARY KEY;
31| INSERT INTO Enfants (Prenom, Sexe, CodeFamille) VALUES ('Alex', 'F', '12');
32| INSERT INTO Enfants (Prenom, Sexe, CodeFamille) VALUES ('Mila', 'F', '2');
33| INSERT INTO Enfants (Prenom, Sexe, CodeFamille) VALUES ('Amandine', 'F', '5');
34| SELECT familles.Nom , Enfants.Prenom FROM familles, Enfants WHERE ( Enfants.CodeFamille = familles.CodeFamille );
35| SELECT familles.nom, familles.PrenomPere FROM familles, Enfants;
36| SELECT ville, min(nb_enfants) as 'Nb enfants minimum', max(nb_enfants) as 'Nb enfants maximum', avg(nb_enfants) as 'Nb enfants moyen' from familles GROUP BY Ville;
37| SELECT * FROM familles ORDER BY nom;
38| SELECT * FROM familles GROUP BY ville, Nom;
39| UPDATE familles SET Adresse2 = '-';
40| UPDATE familles SET Adresse2 = '___',nb_enfants=2;

```

```
41 UPDATE familles SET nb_enfants=4 WHERE Ville = 'Paris';
42 UPDATE familles SET nb_enfants=7 LIMIT 4;
43 UPDATE familles SET nb_enfants=8 ORDER BY Nom LIMIT 3;
44 DELETE FROM familles WHERE Nom = 'Dupont';
45 CREATE TABLE loisirs( Nom VARCHAR( 30 ) NOT NULL );
46 DROP TABLE loisirs;
47 INSERT INTO familles (Nom, PrenomPere) VALUES ('Alouet','Jean'),('Rahtmi','Patrick');
48 INSERT INTO familles (PrenomPere) SELECT Enfants.prenom FROM Enfants;
49 ALTER TABLE familles ADD Sports VARCHAR(50) NOT NULL;
50 ALTER TABLE familles ADD INDEX (Sports);
51 ALTER TABLE familles ADD INDEX (PrenomPere), ADD EmailPere VARCHAR( 50 ) NOT NULL;
52 ALTER TABLE familles CHANGE PrenomPere Prenom_Pere VARCHAR( 45 ) NOT NULL;
53 ALTER TABLE familles DROP Sports;
54 ALTER TABLE familles DROP INDEX PrenomPere;
55 ALTER TABLE familles DISABLE KEYS;
56 ALTER TABLE Enfants RENAME familles_enfants;
57 ALTER TABLE familles ORDER BY Prenom_Pere;
58 ALTER TABLE familles ADD Commentaire LONGTEXT NOT NULL ;
59 ALTER TABLE familles ADD FULLTEXT (Commentaire);
60 UPDATE familles SET Commentaire = 'La vie est un long fleuve tranquille' WHERE CodeFamille = '11';
61 UPDATE familles SET Commentaire = 'Paris, capitale de la France est traversée par un fleuve' WHERE CodeFamille ='4';
62 UPDATE familles SET Commentaire = 'Le ruisseau se jette dans la rivière qui se jette dans le FLEUVE' WHERE CodeFamille = '6';
63 SELECT * FROM familles WHERE MATCH (Commentaire) AGAINST ('fleuve');
64 SELECT Nom, Commentaire FROM familles WHERE MATCH (Commentaire) AGAINST ('-capitale+fleuve'IN BOOLEAN MODE);
```

<html>

Copyright © 2004-2017 Hugh Norris.<br><br> <a rel="license" href="http://creativecommons.org/licenses/by-nc-nd/3.0/fr/"></a><br />Ce(tte) oeuvre est mise à disposition selon les termes de la <a rel="license" href="http://creativecommons.org/licenses/by-nc-nd/3.0/fr/">Licence Creative Commons Attribution

- Pas d'Utilisation Commerciale - Pas de Modification 3.0 France</a>.

</html>

---

From:

<https://www.ittraining.team/> - **www.ittraining.team**



Permanent link:

<https://www.ittraining.team/doku.php?id=elearning:workbooks:french:14:102:l102>

Last update: **2020/01/30 03:27**