

# Révision du langage SQL (grâce au SGBD MariaDB)



## *Préparation des requêtes SQL-DML pour l'application Web*

### Objectifs

Etre capable, à partir du schéma relationnel de la base de données réalisé pendant la modélisation, de préparer le jeu de requêtes en langage SQL-DML permettant d'interroger la base Maria DB.

### Insertion des lignes dans les tables (insertion du jeu de données professionnel)

1. Insertion de lignes dans les tables des comptes et des profils et vérification de la bonne traduction (ou non) de l'association « un-un » (ou de l'héritage) du diagramme UML de classes de la base de données

L'étape d'insertion du jeu de données est primordiale car elle permet de vérifier une fois de plus la structure de sa base de données. Il n'est pas rare, en faisant ses insertions de découvrir des problèmes de conception qu'il faut alors corriger sans tarder !

### ACTIVITES :




- En utilisant le code SQL-DDL des 3 petites bases données en exemple sur Moodle, créez provisoirement les tables proposées dans votre base de données et à l'aide d'insertion de nouvelles lignes dans les tables **vérifiez la bonne traduction, ou non, de l'association « un-un » entre la classe des comptes et celle des profils**. Expliquez votre façon de procéder !
- **Vous vérifierez alors votre propre base !**
- Une fois les tests terminés, vous supprimerez ces tables de test !

**A NOTER :** pour la suite des activités, on utilisera la petite base de données de test n°3.

## 2. Insertion de lignes dans les tables des comptes et des profils

### ACTIVITES :

- 
- **donnez les requêtes SQL pour insérer des lignes** dans les tables permettant la gestion des profils et des comptes des utilisateurs. Faites d'abord l'insertion d'un profil puis d'un compte, et inversement. Que constatez-vous ? → Conseil : par la suite, il faudra conserver les requêtes exécutées et testées !
  - Exécutez le code SQL suivant : **SELECT SHA2('toto1234',256);**  
A votre avis, que donne l'exécution de ce code SQL ?  
→ Conseil : conservez ce code SQL !
  - Utilisez le code SQL testé précédemment pour insérer un nouveau compte !  
→ Conseil : conservez cette requête !
  - Une fois le jeu de données professionnel demandé **page 4 du cahier des charges** complété, **il ne faudra pas oublier d'exporter le code SQL et le jeu de données ajouté (dans un fichier texte d'extension .sql)** pour pouvoir avoir une copie de votre base de données sur votre ordinateur personnel.

→ Rappel : toutes vos tables doivent contenir au minimum 2 ou 3 lignes...

### Manipulations CRUD sur une seule table (en SQL-DML)

Une fois la structure de votre base de données (relations, attributs et clés) créée à l'aide du SGBD MariaDB, il est alors possible d'effectuer des manipulations sur les **données** de la base via les ordres SQL-DML **INSERT** (insertion), **UPDATE** (modification), **DELETE** (suppression) et **SELECT** (extraction).

Pour effectuer les manipulations sur les données des tables de la base de données, il faut préparer des **requêtes** SQL (qui décrivent les opérations que le SGBD doit réaliser).

Il s'agit dans un premier temps de réaliser des manipulations **CRUD** sur la table de gestion des profils : **C** : Create / **R** : Retrieve / **U** : Update / **D** : Delete.

#### Requêtes de gestion (CRUD) des profils des utilisateurs :

A l'aide du langage SQL-DML, préparez vos requêtes :

1. **insérez un compte puis le profil associé en une seule manipulation !**
2. **Modifiez** la validité d'un profil connaissant le pseudo de l'utilisateur.
3. **Supprimez** une ligne de la table des profils connaissant le pseudo d'un utilisateur.
4. **a) Listez** tous les noms, prénoms et statuts des utilisateurs suivant l'ordre alphabétique des noms de famille.  
**b) Listez** tous les noms, prénoms et statuts des utilisateurs rassemblés par statut.
5. **Listez** tous les noms, prénoms et pseudo (ou @email) des utilisateurs de statut 'O' (Organisateurs) dans l'ordre alphabétique décroissant des prénoms.

**Autres requêtes :**

**A l'aide des opérateurs arithmétiques et des fonctions fournies ci-dessous :**

6. listez le nom et le prénom des profils ajoutés en 2022,
7. insérer la date du jour lors de l'ajout d'un profil,
8. déterminez le numéro du dernier profil ajouté (+ nom / prénom ?),
9. déterminez tous les profils ajoutés entre 2 dates à spécifier,
10. dénombrez les Organismes puis les membres du Jury,
11. déterminez les différents statuts d'utilisateurs qui existent (2 actuellement).

→ Fonctions MariaDB à utiliser :

Fonction MariaDB	Description
CURDATE() NOW()	Retourne la date courante au format 'AAAA-MM-JJ' (DATE) ou AAAAMMJJ (INT)
YEAR(date)	Retourne l'année de la date spécifiée
COUNT({*   [DISTINCT] expr })	Retourne le nombre de valeurs de l'expression expr parmi les lignes trouvées par un SELECT
MAX([DISTINCT] expr)	Retourne la valeur maximale de expr
MIN([DISTINCT] expr)	Retourne la valeur minimale de expr (ex : SELECT etu_nom, MIN(etu_note) FROM ETUDIANTS GROUP BY etu_nom; )
DISTINCT	Supprime les doublons dans le résultat obtenu
SHA2(str,hash_len)	Calcule l'empreinte SHA2 de longueur hash_len de la chaîne str



**12. Donnez la requête permettant de vérifier qu'un pseudo et le mot de passe associé existent dans la base de données. A quoi sert cette requête ?**

## Manipulations sur plusieurs tables (en SQL-DML) - Jointures

**Requêtes de gestion des données des utilisateurs :**

13. Modifiez le mot de passe correspondant à un utilisateur connaissant son nom et son prénom.
14. Supprimez **toutes** les données associées à l'utilisateur de pseudo « **mdurand** ». Que faudra-t-il faire si cet utilisateur a ajouté des données (actualités, messages, ...) ?
15. Proposez une requête permettant de vérifier s'il y a bien autant de comptes que de profils dans la base de données.
16. Supprimez tous les comptes n'ayant pas de profil associé.

## ANNEXES

**Annexe 1 : SQL-DML (Structured Query Language – Data Manipulation Language)**

**SQL-DML (« Structured Query Language – Data Manipulation Language »)** : c'est un sous-langage du SQL qui permet l'extraction et la modification de données. Contrairement au SQL-DDL qui travaille sur les structures, le SQL-DML travaille sur les **données** de la base.

*Manuel MariaDB :*

<https://mariadb.com/kb/en/library/data-manipulation/>

INSERT	
<i>Insertion des données des lignes d'une table (INSERT INTO)</i>	
<b>INSERT INTO [ONLY] &lt;table_ou_vue&gt; [&lt;liste_colonnes&gt;]</b> <b>{VALUES &lt;liste_valeur&gt;   DEFAULT VALUES   &lt;requete_select&gt;}</b>	
Exemple	Description
<b>INSERT INTO CLIENT</b> <b>VALUES(7,'jmartin@yahoo.fr','M.','MARTIN','Jean','Rennes');</b>	Insère dans la table CLIENT une ligne (enregistrement) constituée des valeurs 7, 'jmartin@yahoo.fr', 'M.', 'MARTIN', 'Jean', 'Rennes'.
<b>INSERT INTO UTILISATEUR (UTI_ID, UTI_MAIL, UTI_ADR)</b> <b>VALUES(7,'jmartin@yahoo.fr',NULL);</b>	Insère dans une table UTILISATEUR une ligne (enregistrement) avec uniquement les valeurs pour les colonnes spécifiées, ici UTI_ID et UTI_MAIL et <b>aucune valeur pour la colonne UTI_ADR (marqueur NULL)</b> .
<b>INSERT INTO UTILISATEUR (UTI_PSEUDO, UTI_MOT_PASSE)</b> <b>VALUES('titi',MD5('titi123456'));</b>	Insère dans une table UTILISATEUR une ligne (enregistrement) avec uniquement les valeurs pour les colonnes spécifiées ici UTI_PSEUDO et UTI_MOT_PASSE <b>avec appel de la fonction MD5()</b> .
DELETE	
<i>Suppression des données des lignes d'une table (DELETE FROM)</i>	
<b>DELETE [FROM] [ONLY] &lt;table_ou_vue&gt;</b> <b>[WHERE &lt;prédicat&gt;]</b>	
Exemple	Description
<b>DELETE FROM PANIER;</b>	Supprime toutes les lignes (enregistrements) de la table PANIER. <b>DELETE FROM `PANIER`;</b>
<b>DELETE FROM CLIENT</b> <b>WHERE CL_ID=7;</b>	Supprime la ligne d'identifiant 7 de la table CLIENT. <i>Exemple sous MariaDB :</i> <b>DELETE FROM `t_client_cli`</b> <b>WHERE `t_client_cli`.`cli_ID` = 7 ;</b>

<b>DELETE FROM</b> CLIENT <b>WHERE</b> CL_ID <b>BETWEEN</b> 7 <b>AND</b> 10;	Supprime les lignes dont les identifiants sont compris entre 7 et 10 dans la table CLIENT.
<b>UPDATE</b>	
<b>Modification des données des lignes d'une table (UPDATE)</b>	
<b>UPDATE</b> [ONLY] <table_ou_vue> <b>SET</b> {col1=valeur1 [col2=valeur2 [, ...]]   <b>ROW</b> =<ligne_valuée>} [WHERE <prédicat>]	
Exemple	Description
<b>UPDATE</b> CLIENT <b>SET</b> CL_ADRESSE='Brest' <b>WHERE</b> CL_ID=7 ;	Modifie la colonne CL_ADRESSE pour la ligne du client qui a l'identifiant 7.
<b>UPDATE</b> CLIENT <b>SET</b> <b>ROW</b> = <b>ROW</b> (7,'jeanmartinez@yahoo.fr','M.','MARTINEZ','Jean','Brest') <b>WHERE</b> CL_ID=7 ;	Modifie toutes les colonnes de la ligne du client qui a l'identifiant 7. <u>Sous MariaDB :</u> utiliser la syntaxe présentée ligne précédente permettant la modification des colonnes séparément.
<b>SELECT</b>	
<b>Extraction des données d'une table (SELECT FROM)</b>	
<b>SELECT</b> [ALL   DISTINCT] <liste_attributs> <b>FROM</b> {nom_table   nom_vue} [WHERE <prédicat_filtre_données>] [GROUP BY <liste_expressions_sous_ensemble>] [HAVING <prédicat_filtre_resultat>] [WINDOW <paramètre de fenêtrage>] [ORDER BY <liste_expression_de_tri>]	
<u>A noter :</u> à part <b>SELECT FROM</b> , les autres clauses du SELECT sont optionnelles.	
Exemple	Description
<b>SELECT</b> * <b>FROM</b> CLIENT;	<b>Projection (clause SELECT) :</b> On requiert toutes les colonnes pour toutes les lignes de la table CLIENT
<b>SELECT DISTINCT</b> CL_NOM <b>FROM</b> CLIENT;	<b>Projection (clause SELECT) :</b> On requiert uniquement les noms distincts apparaissant dans toutes les lignes de la table CLIENT
<b>SELECT</b> * <b>FROM</b> CLIENT <b>WHERE</b> CL_ADRESSE='brest';	<b>Restriction (clause WHERE) :</b> On recherche les lignes de la table pour lesquels la colonne CL_ADRESSE contient 'brest', c'est à dire les clients habitant à Brest
<b>SELECT</b> CL_ID, CL_NOM <b>FROM</b> CLIENT <b>WHERE</b> CL_PRENOM <b>NOT NULL</b> <b>AND</b> CL_PRENOM <b>LIKE</b> '%A%';	<b>Restriction (clause WHERE) :</b> On recherche les lignes de la table pour lesquels la colonne CL_PRENOM est renseignée (NOT NULL) et contient le caractère 'A'

<b>SELECT *</b> <b>FROM</b> CLIENT <b>ORDER BY</b> CL_NOM;	<b>Tri (clause ORDER BY) :</b> On recherche toutes les lignes de la table CLIENT classées dans l'ordre croissant des noms des clients (ordre alphabétique)
<b>SELECT</b> COUNT(*), CL_ADRESSE <b>FROM</b> CLIENT <b>GROUP BY</b> CL_ADRESSE;	<b>Groupe (clause GROUP BY) :</b> On recherche le nombre de clients habitant toutes les villes mentionnées dans la table
<b>SELECT</b> COUNT(*), CL_ADRESSE <b>FROM</b> CLIENT <b>GROUP BY</b> CL_ADRESSE <b>HAVING</b> COUNT(*) > 10;	<b>Filtrage (clause HAVING) :</b> On recherche les villes dans lesquels plus de 10 clients habitent
<b>SELECT *</b> <b>FROM</b> COMMANDE <b>INNER JOIN</b> CLIENT <b>ON</b> COMMANDE.CL_ID = CLIENT.CL_ID;	<b>Jointure interne (INNER JOIN) :</b> Par défaut, la jointure de 2 tables est dite <b>interne</b> car elle fait correspondre des lignes des 2 tables en fonction d'un prédicat appliqué aux colonnes. Ici, on recherche les commandes réalisées par les clients
<b>SELECT *</b> <b>FROM</b> COMMANDE <b>RIGHT OUTER JOIN</b> CLIENT <b>ON</b> COMMANDE.CL_ID = CLIENT.CL_ID;	<b>Jointures externes (LEFT, RIGHT OUTER JOIN) :</b> La jointure externe permet d'extraire toutes les lignes d'une table en correspondance <b>ou non</b> avec l'autre table. Ici, on recherche tous les clients qui ont passé ou non une commande.
<b>SELECT *</b> <b>FROM</b> CLIENT <b>LEFT OUTER JOIN</b> COMMANDE <b>ON</b> COMMANDE.CL_ID = CLIENT.CL_ID;	<b>Jointures externes (LEFT, RIGHT, OUTER JOIN) :</b> Ici, on recherche tous les clients qui ont passé ou non une commande. <b>A noter :</b> on constate que la table dont la position correspond aux mots clés LEFT ou RIGHT voit <b>toutes ses lignes listées</b> dans la réponse
<b>SELECT *</b> <b>FROM</b> COMMANDE <b>JOIN</b> CLIENT <b>ON</b> COMMANDE.CL_ID = CLIENT.CL_ID;	<b>Équijointure :</b> Une équijointure est une jointure dans laquelle la comparaison se fait à l'aide de l'opérateur d'égalité
<b>SELECT *</b> <b>FROM</b> COMMANDE <b>NATURAL JOIN</b> CLIENT	<b>Jointure naturelle (NATURAL JOIN) :</b> Une jointure naturelle est possible quand les tables possèdent des colonnes et des types identiques

## Annexe 2 : MariaDB – les opérateurs

MariaDB propose différents opérateurs arithmétiques et logiques utiles pour l'écriture des requêtes.

*Manuel MariaDB* : <https://mariadb.com/kb/en/library/operators/>

Opérateurs MariaDB	Description
AND, &&	ET logique
XOR	XOR (OU exclusif) logique
, OR	OU logique
=	Opérateur d' <b>affectation</b> dans la clause SET
=	Opérateur de comparaison (égalité)
:=	Opérateur d'affectation lors de l'utilisation de <b>variables</b>
!=, <>	Opérateur de comparaison (inégalité)
NOT, !	Négation logique
~	Inversion de chaque bit
	OU bit à bit
^	XOR (OU exclusif) bit à bit
&	ET bit à bit
[NOT] BETWEEN ... AND ...	Vérification de l'appartenance (ou non) d'une valeur à un intervalle
IN	Vérifie si une valeur est dans une liste
>, >=	Opérateurs de comparaison <ul style="list-style-type: none"> <li>supérieur et supérieur ou égal</li> </ul>
<, <=	Opérateurs de comparaison <ul style="list-style-type: none"> <li>inférieur et inférieur ou égal</li> </ul>
<<, >>	Opérateur de décalage bit à bit (à gauche << et à droite >>)
[NOT] EXISTS	Est utilisé pour vérifier si une sous-requête (un SELECT dans la clause WHERE) retourne ou non des résultats
IS [NOT] NULL	Vérificateur de la présence d'une valeur, ou non
IS [NOT]	Opérateur de comparaison logique (booléen)
[NOT] LIKE	Compare des chaînes de caractères à une expression (ex : ... <b>WHERE</b> CL_nom <b>LIKE</b> ( '%A%' ); filtre les noms de client contenant un A) <i>A noter :</i> _ désigne n'importe quel caractère et % désigne n'importe quelle séquence de caractères
-	Opérateur de soustraction, opérateur binaire et unaire (change le signe d'un argument)
+	Opérateur d'addition
*	Opérateur de multiplication
/	Opérateur de division
%, MOD	Opérateur modulo
[NOT] REGEXP	Effectue une recherche de chaîne à partir d'une expression régulière
RLIKE	Synonyme de REGEXP
SOUNDS LIKE	Vérifie si deux textes se prononcent de la même manière
DISTINCT	Supprime les doublons dans le résultat obtenu (ex : <b>SELECT DISTINCT</b> CL_nom <b>FROM</b> CLIENT retourne les noms de famille différents présents dans la table CLIENT)

### Annexe 3 : MariaDB – les fonctions

Le SGBD MariaDB offre un grand nombre de fonctions que l'on peut utiliser dans les clauses **SELECT** et **WHERE** des requêtes.

On peut combiner les appels de fonctions.

Manuel SQL en ligne : <http://sql.sh>

### Fonctions de gestion des heures et des dates :

Manuel MariaDB : <https://mariadb.com/kb/en/library/date-time-functions/>

Fonction MariaDB	Description
ADDDATE(date,n)	Ajoute n jours à une date de type DATE ou DATETIME
ADDTIME(date1,date2)	Ajoute les deux dates
CURDATE()	Retourne la date courante au format 'AAAA-MM-JJ' (DATE) ou AAAAMMJJ (INT)
CURRENT_DATE(), CURRENT_DATE	Synonymes de CURDATE()
CURTIME()	Retourne l'heure courante au format 'HH:MM:SS' (DATE) ou HHMMSS (INT)
CURRENT_TIME(), CURRENT_TIME	Synonymes de CURTIME()
CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP	Synonymes de NOW()
DATE_ADD(date,INTERVAL expr type)	Ajoute un intervalle à une date expr désigne un intervalle type indique un format
DATE_FORMAT(date,format)	Présente une date selon le format spécifié
DATE_SUB(date,INTERVAL expr type)	Soustrait un intervalle à une date expr désigne un intervalle type indique un format
DATE(datetime)	Extrait une date à partir d'une expression de type DATETIME
DATEDIFF(date1,date2)	Détermine un nombre entier de jours entre 2 dates
DAYOFMONTH()	Retourne le n° du jour dans le mois (0-31)
DAY(date)	Synonyme de DAYOFMONTH()
DAYNAME(date)	Retourne le nom du jour (en anglais)
DAYOFYEAR(date)	Retourne le n° du jour dans l'année (1-366)
EXTRACT(type FROM date)	Extrait une partie d'une date selon un type d'intervalle
FROM_DAYS(n)	Retourne une date à partir d'un nombre de jours
FROM_UNIXTIME()	Retourne une date à partir d'un nombre de jours depuis le 01/01/1970
GET_FORMAT(date,format)	Convertit une valeur de type date, time ou datetime dans le format spécifié (ex : GET_FORMAT(DATE,'EUR') donne une date sous forme '%d %m.%Y' )



HOUR(time)	Extrait l'heure d'un temps
LAST_DAY(date)	Retourne le dernier jour du mois d'une date
LOCALTIME(), LOCALTIME	Synonymes de NOW()
LOCALTIMESTAMP, LOCALTIMESTAMP()	Synonymes de NOW()
MAKEDATE(annee,nbjour)	Crée une date à partir de l'année et d'un nombre de jours
MAKETIME(heure,minute, seconde)	Crée une heure à partir d'un nombre d'heures, minutes et secondes
MICROSECOND(date)	Extrait les micro-secondes d'une date
MINUTE(time)	Extrait les minutes d'un temps
MONTH(date)	Retourne le n° du mois d'une date
MONTHNAME(date)	Retourne le nom du mois d'une date
NOW()	Retourne la date et l'heure courantes au format 'AAAA-MM-JJ HH:MM:SS' (DATE) ou AAAAMMJJ HHMMSS (INT)
PERIOD_DIFF(int1,int2)	Retourne le nombre de mois séparant deux dates au format AAMM ou AAAAMM (INT)
SEC_TO_TIME(secondes)	Convertit des secondes en format 'HH:MM:SS'
SECOND(time)	Extrait le nombre de secondes d'un temps (0-59)
STR_TO_DATE(chaine,format)	Convertit une chaîne en date
SUBDATE(date,n)	Retranche n jours à une date
SUBTIME(date1,date2)	Retranche deux dates
SYSDATE()	Retourne une date courante d'exécution au format 'AAAA-MM-JJ HH:MM:SS' (DATE) ou AAAAMMJJ HHMMSS (INT)
TIME_FORMAT(time,format)	Convertit une heure dans le format spécifié
TIME_TO_SEC(time)	Retourne le nombre de secondes correspondant au temps spécifié
TIME(datetime)	Extrait le temps d'une date de type DATETIME
TIMEDIFF(date1,date2)	Donne le temps entre deux dates
TIMESTAMP(date)	Retourne <ul style="list-style-type: none"> <li>l'expression de la date (si un seul paramètre passé)</li> <li>la somme des paramètres (si 2 paramètres passés)</li> </ul>
TIMESTAMPADD(intervalle,int,date)	Ajoute à la date (de type datetime) un intervalle
TIMESTAMPDIFF(intervalle,int,date)	Retranche à la date (de type datetime) un intervalle
TO_DAYS(date)	Retourne le nombre de jours correspondant à la date spécifiée
UNIX_TIMESTAMP(date)	Retourne le nombre de secondes depuis le 01/01/1970 et jusqu'à la date spécifiée
UTC_DATE()	Retourne la date courante UTC (par rapport au méridien de Greenwich)
UTC_TIME()	Retourne l'heure courante UTC (par rapport au méridien de Greenwich)
UTC_TIMESTAMP()	Retourne la date et l'heure courantes UTC (par rapport au méridien de Greenwich)
WEEK(date)	Retourne le n° de la semaine d'une date
WEEKDAY()	Retourne l'index du jour dans la semaine (0 pour lundi, 1 pour mardi,...)
WEEKOFYEAR()	Retourne le n° de la semaine en cours (1-53)
YEAR(date)	Retourne l'année de la date spécifiée
YEARWEEK(date)	Retourne l'année et la semaine de la date spécifiée

## Fonctions mathématiques :

Manuel MariaDB : <https://mariadb.com/kb/en/library/numeric-functions/>

Fonction MariaDB	Description
ABS(n)	Retourne la valeur absolue de n
ACOS(n)	Retourne l'arc cosinus de n en radians
ASIN(n)	Retourne l'arc sinus de n en radians
ATAN(n)	Retourne l'arc tangente de n en radians
CEIL(n)	Retourne le plus petit entier supérieur ou égal à n
CONV(n,from_base,to_base)	Convertit un nombre n d'une base vers une autre base de numération et retourne une chaîne
COS(n)	Exprime le cosinus de n en radians
COT(n)	Exprime la cotangente de n en radians
CRC32(expr)	Calcule le code de redondance cyclique CRC32 et renvoie une valeur de 32 bits non signée
DEGREES(n)	Convertit des radians en degrés
EXP(n)	Calcule la valeur de e puissance n
FORMAT(nombre,n)	Retourne une chaîne correspondant au nombre formaté avec n décimales
FLOOR(n)	Retourne le plus grand entier inférieur ou égal à n
LN(n)	Retourne le logarithme népérien de n
LOG10(n)	Retourne le logarithme en base 10 de n
LOG2(n)	Retourne le logarithme en base 2 de n
LOG(m,n)	Retourne le logarithme de n en base m
MOD(m,n)	Retourne le reste de la division entière de m par n
PI()	Retourne la valeur de pi
POW(m,n)	Calcule m à la puissance n
RADIANS(n)	Convertit des degrés en radians
RAND()	Valeur de type float aléatoire (à 14 décimales) entre 0 et 1
ROUND(m,n)	Arrondit la valeur m au nombre n de décimales spécifié
SIGN(n)	Retourne le signe d'un nombre
SIN(n)	Retourne le sinus de n exprimé en radians
SQRT(n)	Calcule la racine carrée de n
TAN(n)	Retourne la tangente de n exprimée en radians
TRUNCATE(n,m)	Coupe la valeur de n à m décimales

**Fonctions de gestion des chaînes :**

Manuel MariaDB : <https://mariadb.com/kb/en/library/string-functions/>

Fonction MariaDB	Description
ASCII(c)	Retourne la valeur numérique du code ASCII du caractère spécifié
BIN(n)	Fonction de conversion qui convertit <b>n</b> en valeur binaire (ex : BIN(12) retourne '1100')
BIT_LENGTH(ch)	Retourne la taille de la chaîne en nombre de bits
CHAR_LENGTH(ch)	Retourne le nombre de caractères de la chaîne
CHAR(n)	Retourne le caractère correspondant à la valeur numérique spécifiée dans le jeu de caractères actuel
CHARACTER_LENGTH()	Synonyme de CHAR_LENGTH()
CONCAT_WS(separateur,c1,c2,...)	Retourne une chaîne constituée de la concaténation des chaînes séparées par le séparateur précisé
CONCAT(c1,c2)	Retourne la chaîne correspondant à la concaténation des deux chaînes spécifiées
FIELD(c,c1,c2,...)	Retourne l'index (position) correspondant à la première égalité entre c et c1, c et c2... et 0 si aucune égalité n'est trouvée
HEX(c)	Retourne une représentation hexadécimale d'une valeur décimale ou d'une chaîne
INSERT(c1,pos,nb,c2)	Insère nb caractères de la séquence de caractères c2 dans la chaîne c1 à partir de la position pos
INSTR(c1,c2)	Retourne l'index (position) de l'emplacement de la séquence de caractères c2 dans la chaîne c1
LCASE()	Synonyme de LOWER()
LEFT(ch,n)	Extrait les n premiers caractères de la chaîne à partir de la gauche
LENGTH(ch)	Retourne la taille de la chaîne en octets
LOAD_FILE(nf)	Charge le fichier dont le nom est passé en paramètre
LOCATE(c2,c1)	Retourne la position de la première occurrence de la chaîne c2 dans la chaîne c1
LOWER(ch)	Renvoie la chaîne spécifiée en minuscules
LPAD(c1,n,c2)	Insère à gauche de la chaîne c1 la séquence de caractères c2 sur n caractères
MID(ch,n)	Retourne la séquence de caractères à partir de la position spécifiée
OCT(n)	Retourne une chaîne contenant la représentation octale d'un nombre
OCTET_LENGTH()	Synonyme de LENGTH()
POSITION()	Synonyme de LOCATE()
REPEAT(ch,n)	Répète une chaîne le nombre de fois spécifié
REPLACE(c1,c2,c3)	Remplace les occurrences de la chaîne c2 dans la chaîne c1 par la chaîne c3
REVERSE(ch)	Renverse la chaîne
RIGHT(ch,n)	Extrait les n caractères de la chaîne à partir de la droite

RPAD(c1,n,c2)	Ajoute n fois à la fin de c1 (à droite) la chaîne c2
SOUNDEX(c)	Extrait la phonétique d'une chaîne (en anglais)
SPACE(n)	Retourne une chaîne formée de n espaces
STRCMP(c1,c2)	Compare deux chaînes
SUBSTR(ch,n,[nbcар])	Extrait une séquence de caractères de la chaîne ch à partir de la position n et sur nbcар caractères
SUBSTRING_INDEX(ch,delim,n)	Retourne une partie de la chaîne ch située avant le nombre spécifié d'apparition du caractère délimiteur (ex : SUBSTRING_INDEX( 'www.mysql.com', '.', 2); donne www.mysql car c'est la portion de la chaîne située avant le 2ème . )
SUBSTRING(ch,pos)	Retourne une partie de la chaîne à partir de la position spécifiée
TRIM(c1 FROM c2)	Enlève les caractères c1 de la chaîne c2
UCASE()	Synonyme de UPPER()
UPPER()	Renvoie la chaîne en majuscules

## Fonctions d'encodage des chaînes :

Manuel MariaDB : <https://mariadb.com/kb/en/library/encryption-hashing-and-compression-functions/>

Fonction MariaDB	Description
AES_DECRYPT(ch)	Déchiffre une chaîne en utilisant l'algorithme AES
AES_ENCRYPT(ch)	Chiffre une chaîne en utilisant l'algorithme AES
COMPRESS()	Compresse une chaîne et retourne le résultat sous forme binaire
DECODE(ch)	Décode une chaîne encodée grâce à ENCODE()
ENCODE(ch,pass)	Encode une chaîne en utilisant le mot de passe pass et retourne une chaîne de même longueur que ch
MD5(ch)	Calcule un checksum (somme de test) MD5
SHA1(), SHA()	Calcule un checksum (somme de test) SHA-1 de 160 bits
SHA2()	Calcule un checksum (somme de test) SHA-2
UNCOMPRESS(ch)	Décompresse une chaîne compressée
UNCOMPRESSED_LENGTH(ch)	Retourne la longueur de la chaîne avant compression
VALIDATE_PASSWORD_STRENGTH()	Détermine la force du mot de passe

**Fonctions d'information :**

Manuel MariaDB : <https://mariadb.com/kb/en/library/information-functions/>

Fonction MariaDB	Description
BENCHMARK(n,expr)	Exécute n fois une expression
CHARSET(ch)	Retourne le jeu de caractères de la chaîne spécifiée (ex : CHARSET( ' abc ' ) ; peut donner 'latin1')
COLLATION()	Retourne la collation de la chaîne spécifiée (ex : COLLATION( ' abc ' ) ; peut donner 'latin1_swedish_ci')
CONNECTION_ID()	Retourne l'ID de connexion
CURRENT_USER(), CURRENT_USER	Retourne l'utilisateur courant et le nom de l'hôte
DATABASE()	Retourne le nom de la base courante
FOUND_ROWS()	Retourne le nombre de lignes qui serait retourné sans la clause LIMIT dans un SELECT
LAST_INSERT_ID()	Valeur de l'AUTO-INCREMENT lors du dernier INSERT
ROW_COUNT()	Retourne le nombre de lignes mises à jour
SCHEMA()	Synonyme de DATABASE()
SESSION_USER()	Synonyme de USER()
SYSTEM_USER()	Synonyme de USER()
USER()	Retourne le nom d'utilisateur et le nom d'hôte fournis par le client
VERSION()	Retourne une chaîne indiquant la version du serveur MariaDB

**Fonctions de groupe (« GROUP BY ») :**

Manuel MariaDB : <https://mariadb.com/kb/en/library/aggregate-functions/>

Fonction MariaDB	Description
AVG(expr)	Retourne la moyenne de l'expression spécifiée
BIT_AND(expr)	Retourne le résultat du ET bit à bit appliqué à l'ensemble des bits d'expr
BIT_OR(expr)	Retourne le résultat du OU bit à bit appliqué à l'ensemble des bits d'expr
BIT_XOR(expr)	Retourne le résultat du OU exclusif bit à bit appliqué à l'ensemble des bits d'expr
COUNT({*   [DISTINCT] expr })	Retourne le nombre de valeurs de l'expression expr parmi les lignes trouvées par un SELECT
MAX([DISTINCT] expr)	Retourne la valeur maximale de expr
MIN([DISTINCT] expr)	Retourne la valeur minimale de expr (ex : SELECT etu_nom, MIN(etu_note) FROM ETUDIANTS GROUP BY etu_nom; )
STDDEV(expr)	Retourne l'écart-type de expr
SUM([DISTINCT] expr)	Retourne la somme de expr
VARIANCE(expr)	Retourne la variance de expr