



**eBook Gratuit**

# APPRENEZ

---

# MySQL

eBook gratuit non affilié créé à partir des  
**contributeurs de Stack Overflow.**

**#mysql**

# Table des matières

|  |           |
|--|-----------|
| À propos.....  | 1         |
| <b>Chapitre 1: Démarrer avec MySQL.....</b>  | <b>2</b>  |
| Remarques.....   | 2         |
| Versions.....  | 2         |
| Exemples.....  | 3         |
| Commencer.....   | 3         |
| Exemples de schémas d'information.....   | 7         |
| <b>Liste des processus.....</b>  | <b>7</b>  |
| <b>Procédure stockée Recherche.....</b>  | <b>7</b>  |
| <b>Chapitre 2: ALTER TABLE.....</b>  | <b>9</b>  |
| Syntaxe.....   | 9         |
| Remarques.....   | 9         |
| Exemples.....  | 10        |
| Changement de moteur de stockage; reconstruire la table; changez file_per_table..... | 10        |
| ALTER COLONNE DE TABLE.....  | 10        |
| ALTER table ajouter INDEX.....   | 10        |
| Modifier la valeur d'incrémentatation automatique.....                               | 11        |
| Modification du type d'une colonne de clé primaire.....                              | 11        |
| Changer la définition de la colonne.....   | 11        |
| Renommer une base de données MySQL.....  | 12        |
| Echanger les noms de deux bases de données MySQL.....                                | 12        |
| Renommer une table MySQL.....  | 13        |
| Renommer une colonne dans une table MySQL.....                                       | 13        |
| <b>Chapitre 3: Arithmétique.....</b>   | <b>15</b> |
| Remarques.....   | 15        |
| Exemples.....  | 15        |
| Opérateurs arithmétiques.....  | 15        |
| BIGINT.....  | 15        |
| DOUBLE.....  | 16        |
| Constantes Mathématiques.....  | 16        |

|   |           |
|---|-----------|
| Pi.....   | 16        |
| Trigonométrie (SIN, COS).....   | 16        |
| Sinus.....  | 16        |
| Cosinus.....  | 16        |
| Tangente.....   | 17        |
| Arc Cosinus (cosinus inverse).....  | 17        |
| Arc Sine (sinus inverse).....   | 17        |
| Arc tangente (tangente inverse).....  | 17        |
| Cotangente.....   | 17        |
| Conversion.....   | 18        |
| Arrondi (ROUND, FLOOR, CEIL).....   | 18        |
| Arrondir un nombre décimal à une valeur entière.....  | 18        |
| Arrondir un nombre.....   | 18        |
| Arrondissez un nombre.....  | 18        |
| Arrondissez un nombre décimal à un nombre spécifié de décimales.....                            | 19        |
| Augmenter un nombre à une puissance (POW).....  | 19        |
| Racine Carrée (SQRT).....   | 19        |
| Nombres Aléatoires (RAND).....  | 19        |
| Générer un nombre aléatoire.....  | 19        |
| Nombre aléatoire dans une plage.....  | 19        |
| Valeur absolue et signe (ABS, SIGN).....  | 20        |
| <b>Chapitre 4: Changer le mot de passe.....</b>   | <b>21</b> |
| Exemples.....   | 21        |
| Changer le mot de passe root MySQL sous Linux.....  | 21        |
| Modifier le mot de passe root MySQL dans Windows.....   | 22        |
| Processus.....  | 22        |
| <b>Chapitre 5: CHARGER L'INFILE DE DONNÉES.....</b>   | <b>23</b> |
| Syntaxe.....  | 23        |
| Exemples.....   | 23        |
| utiliser LOAD DATA INFILE pour charger une grande quantité de données dans la base de donn..... | 23        |
| Importer un fichier CSV dans une table MySQL.....   | 24        |
| Charger des données avec des doublons.....  | 24        |

|   |           |
|---|-----------|
| <b>CHARGER LES DONNÉES LOCALES</b> .....  | <b>24</b> |
| <b>LOAD DATA INFILE 'fname' REMPLACER</b> .....   | <b>24</b> |
| <b>LOAD DATA INFILE 'fname' IGNORE</b> .....  | <b>25</b> |
| <b>Charger via la table intermédiaire</b> .....   | <b>25</b> |
| import / export.....  | 25        |
| <b>Chapitre 6: Client MySQL</b> .....   | <b>26</b> |
| Syntaxe.....  | 26        |
| Paramètres.....   | 26        |
| Exemples.....   | 26        |
| Connexion de base.....  | 27        |
| Exécuter des commandes.....   | 27        |
| Exécuter la commande à partir d'une chaîne.....   | 27        |
| Exécuter à partir du fichier script:.....   | 28        |
| Ecrire la sortie sur un fichier.....  | 28        |
| <b>Chapitre 7: Codes d'erreur</b> .....   | <b>29</b> |
| Exemples.....   | 29        |
| Code d'erreur 1064: erreur de syntaxe.....  | 29        |
| Code d'erreur 1175: Mise à jour sécurisée.....  | 29        |
| Code d'erreur 1215: Impossible d'ajouter une contrainte de clé étrangère.....                   | 30        |
| 1045 Accès refusé.....  | 31        |
| 1236 "position impossible" dans la réplication.....   | 31        |
| 2002, 2003 Impossible de se connecter.....  | 32        |
| 1067, 1292, 1366, 1411 - Mauvaise valeur pour le nombre, la date, la valeur par défaut, et..... | 32        |
| 126, 127, 134, 144, 145.....  | 32        |
| 139.....  | 33        |
| 1366.....   | 33        |
| 126, 1054, 1146, 1062, 24.....  | 33        |
| <b>Chapitre 8: COMMANDÉ PAR</b> .....   | <b>35</b> |
| Exemples.....   | 35        |
| Les contextes.....  | 35        |
| De base.....  | 35        |
| Ascendant descendant.....   | 35        |

|   |           |
|---|-----------|
| Des astuces.....  | 35        |
| <b>Chapitre 9: Commentaire Mysql.....</b>   | <b>37</b> |
| Remarques.....  | 37        |
| Exemples.....   | 37        |
| Ajouter des commentaires.....   | 37        |
| Définitions des tables de commentaires.....   | 37        |
| <b>Chapitre 10: Configuration de la connexion SSL.....</b>                                  | <b>39</b> |
| Exemples.....   | 39        |
| Configuration pour les systèmes basés sur Debian.....                                       | 39        |
| <b>Génération d'une clé CA et d'une clé SSL.....</b>  | <b>39</b> |
| <b>Ajouter les clés à MySQL.....</b>  | <b>39</b> |
| <b>Tester la connexion SSL.....</b>   | <b>40</b> |
| <b>Application du protocole SSL.....</b>  | <b>40</b> |
| Références et lectures complémentaires:.....  | 41        |
| Configuration pour CentOS7 / RHEL7.....   | 41        |
| <b>Tout d'abord, connectez-vous à dbserver.....</b>   | <b>41</b> |
| <b>FIN DU SERVEUR CÔTÉ TRAVAIL POUR MAINTENANT.....</b>                                     | <b>43</b> |
| <b>toujours sur le client ici.....</b>  | <b>44</b> |
| <b>MAINTENANT, NOUS SOMMES PRÊTS À TESTER LA CONNEXION SÉCURISÉE.....</b>                   | <b>45</b> |
| <b>Nous sommes toujours sur appclient ici.....</b>  | <b>45</b> |
| <b>Chapitre 11: Configuration et réglage.....</b>   | <b>47</b> |
| Remarques.....  | 47        |
| Exemples.....   | 47        |
| Performance InnoDB.....   | 47        |
| Paramètre permettant l'insertion de données volumineuses.....                               | 47        |
| Augmenter la limite de chaîne pour group_concat.....  | 48        |
| Configuration minimale d'InnoDB.....  | 48        |
| Sécurisation du cryptage MySQL.....   | 49        |
| <b>Chapitre 12: Connexion avec UTF-8 en utilisant divers langages de programmation.....</b> | <b>50</b> |
| Exemples.....   | 50        |
| Python.....   | 50        |

|  |           |
|--|-----------|
| PHP .....  | 50        |
| <b>Chapitre 13: Conversion de MyISAM à InnoDB .....</b>                              | <b>52</b> |
| Exemples .....   | 52        |
| Conversion de base .....   | 52        |
| Conversion de toutes les tables dans une base de données .....                       | 52        |
| <b>Chapitre 14: Création de bases de données .....</b>                               | <b>53</b> |
| Syntaxe .....  | 53        |
| Paramètres .....   | 53        |
| Exemples .....   | 53        |
| Créer une base de données, des utilisateurs et des subventions .....                 | 53        |
| MyDatabase .....   | 55        |
| Bases de données système .....   | 56        |
| Création et sélection d'une base de données .....                                    | 56        |
| <b>Chapitre 15: Création de table .....</b>  | <b>58</b> |
| Syntaxe .....  | 58        |
| Remarques .....  | 58        |
| Exemples .....   | 58        |
| Création de table de base .....  | 58        |
| <b>Définition des valeurs par défaut .....</b>                                       | <b>59</b> |
| Création de table avec clé primaire .....  | 59        |
| <b>Définition d'une colonne en tant que clé primaire (définition en ligne) .....</b> | <b>60</b> |
| <b>Définition d'une clé primaire à plusieurs colonnes .....</b>                      | <b>60</b> |
| Création de table avec clé étrangère .....   | 61        |
| Cloner une table existante .....   | 61        |
| CREATE TABLE FROM SELECT .....   | 62        |
| Afficher la structure du tableau .....   | 63        |
| Table Create With TimeStamp Column pour afficher la dernière mise à jour .....       | 63        |
| <b>Chapitre 16: Créer un nouvel utilisateur .....</b>                                | <b>65</b> |
| Remarques .....  | 65        |
| Exemples .....   | 65        |
| Créer un utilisateur MySQL .....   | 65        |

|  |           |
|--|-----------|
| Spécifiez le mot de passe.....   | 65        |
| Créer un nouvel utilisateur et accorder tous les privilèges au schéma.....                             | 65        |
| Renommer l'utilisateur.....  | 66        |
| <b>Chapitre 17: EFFACER.....</b>   | <b>67</b> |
| Syntaxe.....   | 67        |
| Paramètres.....  | 67        |
| Exemples.....  | 67        |
| Supprimer avec la clause Where.....  | 67        |
| Supprimer toutes les lignes d'une table.....   | 68        |
| Supprimer les suppressions.....  | 68        |
| Suppressions multi-tables.....   | 68        |
| <b>clés étrangères.....</b>  | <b>69</b> |
| Suppression de base.....   | 70        |
| SUPPRIMER vs TRUNCATE.....   | 70        |
| Multi-table DELETE.....  | 71        |
| <b>Chapitre 18: ENUM.....</b>  | <b>72</b> |
| Exemples.....  | 72        |
| Pourquoi ENUM?.....  | 72        |
| TINYINT comme alternative.....   | 72        |
| VARCHAR comme alternative.....   | 73        |
| Ajouter une nouvelle option.....   | 73        |
| NULL vs NOT NULL.....  | 73        |
| <b>Chapitre 19: Erreur 1055: ONLY_FULL_GROUP_BY: quelque chose n'est pas dans la clause GROUP.....</b> | <b>75</b> |
| Introduction.....  | 75        |
| Remarques.....   | 75        |
| Exemples.....  | 76        |
| Utiliser et utiliser GROUP BY.....   | 76        |
| Abuser de GROUP BY pour obtenir des résultats imprévisibles: la loi de Murphy.....                     | 76        |
| Mauvaise utilisation de GROUP BY avec SELECT * et comment y remédier.....                              | 77        |
| DE N'IMPORTE QUELLE VALEUR().....  | 78        |
| <b>Chapitre 20: Événements.....</b>  | <b>79</b> |

|  |           |
|--|-----------|
| Exemples.....  | 79        |
| Créer un événement.....  | 79        |
| Schéma pour tester.....  | 79        |
| Créer 2 événements, 1ère course tous les jours, 2ème parcours toutes les 10 minutes..... | 79        |
| Afficher les statuts d'événement (différentes approches).....                            | 80        |
| Des choses aléatoires à considérer.....  | 81        |
| <b>Chapitre 21: Expressions régulières.....</b>  | <b>82</b> |
| Introduction.....  | 82        |
| Exemples.....  | 82        |
| REGEXP / RLIKE.....  | 82        |
| Motif ^.....   | 82        |
| Pattern \$ **.....   | 82        |
| PAS REGEXP.....  | 83        |
| Regex Contient.....  | 83        |
| Tout personnage entre [].....  | 83        |
| Motif ou  .....  | 83        |
| <b>Compter les expressions régulières.....</b>   | <b>83</b> |
| <b>Chapitre 22: Extraire les valeurs du type JSON.....</b>                               | <b>85</b> |
| Introduction.....  | 85        |
| Syntaxe.....   | 85        |
| Paramètres.....  | 85        |
| Remarques.....   | 85        |
| Exemples.....  | 86        |
| Lire la valeur du tableau JSON.....  | 86        |
| Opérateurs d'extraction JSON.....  | 86        |
| <b>Chapitre 23: Fichiers journaux.....</b>   | <b>88</b> |
| Exemples.....  | 88        |
| Une liste.....   | 88        |
| Journal de requête lent.....   | 88        |
| Journal de requête général.....  | 89        |
| Journal des erreurs.....   | 91        |



|  |            |
|--|------------|
| <b>Chapitre 24: Groupage</b> .....   | <b>93</b>  |
| Exemples .....   | 93         |
| La désambiguïisation .....   | 93         |
| <b>Chapitre 25: Index et clés</b> .....                                    | <b>94</b>  |
| Syntaxe .....  | 94         |
| Remarques .....  | 94         |
| <b>Les concepts</b> .....  | <b>94</b>  |
| Exemples .....   | 95         |
| Créer un index .....   | 95         |
| Créer un index unique .....  | 95         |
| Index de baisse .....  | 95         |
| Créer un index composite .....   | 95         |
| Touche AUTO_INCREMENT .....  | 95         |
| <b>Chapitre 26: Informations sur le serveur</b> .....                      | <b>97</b>  |
| Paramètres .....   | 97         |
| Exemples .....   | 97         |
| SHOW VARIABLES exemple .....   | 97         |
| Exemple de SHOW STATUS .....   | 98         |
| <b>Chapitre 27: INSÉRER</b> .....  | <b>99</b>  |
| Syntaxe .....  | 99         |
| Remarques .....  | 99         |
| Exemples .....   | 100        |
| Insert de base .....   | 100        |
| INSERT, ON DUPLICATE MISE À JOUR DE LA CLÉ .....                           | 100        |
| Insertion de plusieurs lignes .....  | 100        |
| Ignorer les lignes existantes .....  | 101        |
| INSERT SELECT (Insérer des données d'une autre table) .....                | 102        |
| INSERT avec AUTO_INCREMENT + LAST_INSERT_ID () .....                       | 102        |
| Identifiants AUTO_INCREMENT perdus .....                                   | 104        |
| <b>Chapitre 28: Installez le conteneur Mysql avec Docker-Compose</b> ..... | <b>106</b> |
| Exemples .....   | 106        |
| Exemple simple avec docker-compose .....                                   | 106        |

|   |            |
|---|------------|
| <b>Chapitre 29: Jeux de caractères et classements</b> .....               | <b>107</b> |
| Exemples.....   | 107        |
| Déclaration.....  | 107        |
| Connexion.....  | 107        |
| Quel jeu de caractères et quelle collection?.....                         | 107        |
| Définition des jeux de caractères sur les tables et les champs.....       | 108        |
| <b>Chapitre 30: JOINS: Rejoignez 3 table avec le même nom d'id.</b> ..... | <b>109</b> |
| Exemples.....   | 109        |
| Rejoignez 3 tables sur une colonne du même nom.....                       | 109        |
| <b>Chapitre 31: Joint</b> .....   | <b>110</b> |
| Syntaxe.....  | 110        |
| Exemples.....   | 110        |
| Exemples de jointure.....   | 110        |
| JOIN avec sous-requête (table "Derived").....                             | 110        |
| Récupérer les clients avec les commandes - variations sur un thème.....   | 111        |
| Full Outer Join.....  | 112        |
| Inner-join pour 3 tables.....   | 113        |
| Jointure visualisée.....  | 114        |
| <b>Chapitre 32: JSON</b> .....  | <b>116</b> |
| Introduction.....   | 116        |
| Remarques.....  | 116        |
| Exemples.....   | 116        |
| Créer un tableau simple avec une clé primaire et un champ JSON.....       | 116        |
| Insérer un simple JSON.....   | 116        |
| Insérer des données mixtes dans un champ JSON.....                        | 116        |
| Mise à jour d'un champ JSON.....  | 117        |
| Données CAST au type JSON.....  | 117        |
| Créer un objet et un tableau Json.....                                    | 117        |
| <b>Chapitre 33: L'optimisation des performances</b> .....                 | <b>119</b> |
| Syntaxe.....  | 119        |
| Remarques.....  | 119        |
| Exemples.....   | 119        |

|   |            |
|---|------------|
| Ajouter le bon index.....   | 119        |
| Définissez le cache correctement.....   | 120        |
| Évitez les constructions inefficaces.....   | 120        |
| Les négatifs.....   | 120        |
| Avoir un INDEX.....   | 120        |
| Ne cache pas dans la fonction.....  | 121        |
| OU.....   | 121        |
| Sous-requêtes.....  | 121        |
| JOIN + GROUP BY.....  | 122        |
| <b>Chapitre 34: Limite et décalage.....</b>   | <b>123</b> |
| Syntaxe.....  | 123        |
| Remarques.....  | 123        |
| Exemples.....   | 123        |
| Relation Limite et Offset.....  | 123        |
| <b>Clause LIMIT avec un argument.....</b>   | <b>123</b> |
| <b>Clause LIMIT avec deux arguments.....</b>  | <b>124</b> |
| <b>Mot clé OFFSET : syntaxe alternative.....</b>                                    | <b>125</b> |
| <b>Chapitre 35: Manipulation des fuseaux horaires.....</b>                          | <b>126</b> |
| Remarques.....  | 126        |
| Exemples.....   | 126        |
| Récupérez la date et l'heure actuelles dans un fuseau horaire particulier.....      | 126        |
| Convertir une valeur stockée `DATE` ou `DATETIME` dans un autre fuseau horaire..... | 127        |
| Récupère les valeurs stockées de TIMESTAMP dans un fuseau horaire particulier.....  | 127        |
| Quel est le paramètre de fuseau horaire local de mon serveur?.....                  | 127        |
| Quelles valeurs time_zone sont disponibles sur mon serveur?.....                    | 128        |
| <b>Chapitre 36: METTRE À JOUR.....</b>  | <b>129</b> |
| Syntaxe.....  | 129        |
| Exemples.....   | 129        |
| Mise à jour de base.....  | 129        |
| Mise à jour d' une ligne.....   | 129        |
| Mise à jour de toutes les lignes.....   | 129        |
| Mise à jour avec le modèle de jointure.....   | 130        |

|  |            |
|--|------------|
| MISE À JOUR avec ORDER BY et LIMIT.....                                | 130        |
| MISE À JOUR de table multiple.....                                     | 131        |
| MISE À JOUR en vrac.....   | 131        |
| <b>Chapitre 37: Moteur MyISAM.....</b>                                 | <b>133</b> |
| Remarques.....   | 133        |
| Exemples.....  | 133        |
| MOTEUR = MyISAM.....   | 133        |
| <b>Chapitre 38: Mots réservés.....</b>                                 | <b>134</b> |
| Introduction.....  | 134        |
| Remarques.....   | 134        |
| Exemples.....  | 139        |
| Erreurs dues aux mots réservés.....                                    | 139        |
| <b>Chapitre 39: MySQL Admin.....</b>                                   | <b>141</b> |
| Exemples.....  | 141        |
| Changer le mot de passe root.....                                      | 141        |
| Déposer la base de données.....  | 141        |
| Atomic RENAME & Rechargement de table.....                             | 141        |
| <b>Chapitre 40: MySQL LOCK TABLE.....</b>                              | <b>142</b> |
| Syntaxe.....   | 142        |
| Remarques.....   | 142        |
| Exemples.....  | 142        |
| Mysql Serrures.....  | 142        |
| Verrouillage au niveau des lignes.....                                 | 143        |
| <b>Chapitre 41: Mysql Performance Tips.....</b>                        | <b>146</b> |
| Exemples.....  | 146        |
| Sélectionner l'optimisation de la déclaration.....                     | 146        |
| Optimisation de la disposition de stockage pour les tables InnoDB..... | 146        |
| Construire un index composite.....                                     | 147        |
| <b>Chapitre 42: mysqlimport.....</b>                                   | <b>149</b> |
| Paramètres.....  | 149        |
| Remarques.....   | 149        |

|  |            |
|--|------------|
| Exemples.....  | 149        |
| Utilisation de base.....   | 149        |
| Utiliser un délimiteur de champ personnalisé.....                    | 150        |
| Utiliser un délimiteur de ligne personnalisé.....                    | 150        |
| Gestion des clés en double.....                                      | 150        |
| Import conditionnel.....   | 151        |
| Importer un csv standard.....  | 151        |
| <b>Chapitre 43: NUL.....</b>   | <b>152</b> |
| Exemples.....  | 152        |
| Utilise NULL.....  | 152        |
| Test NULL.....   | 152        |
| <b>Chapitre 44: Opérations de chaîne.....</b>                        | <b>153</b> |
| Paramètres.....  | 153        |
| Exemples.....  | 155        |
| Rechercher un élément dans une liste séparée par des virgules.....   | 155        |
| STR_TO_DATE - Convertit la chaîne à la date.....                     | 156        |
| LOWER () / LCASE ().....   | 156        |
| REPLACER().....  | 156        |
| SUBSTRING ().....  | 156        |
| UPPER () / UCASE ().....   | 157        |
| LONGUEUR().....  | 157        |
| CHAR_LENGTH ().....  | 157        |
| HEX (str).....   | 157        |
| <b>Chapitre 45: Opérations de date et heure.....</b>                 | <b>158</b> |
| Exemples.....  | 158        |
| À présent().....   | 158        |
| Arithmétique de date.....  | 158        |
| Test par rapport à une plage de dates.....                           | 159        |
| SYSDATE (), NOW (), CURDATE ().....                                  | 159        |
| Extraire la date à partir de la date donnée ou de la date-heure..... | 159        |
| Utilisation d'un index pour une recherche par date et heure.....     | 159        |
| <b>Chapitre 46: Par groupe.....</b>                                  | <b>161</b> |

|   |            |
|---|------------|
| Syntaxe.....  | 161        |
| Paramètres.....   | 161        |
| Remarques.....  | 161        |
| Exemples.....   | 161        |
| GROUPE EN UTILISANT LA Fonction SUM.....                      | 161        |
| Groupe en utilisant la fonction MIN.....                      | 162        |
| GROUPE EN UTILISANT LA FONCTION COUNT.....                    | 162        |
| GROUP BY en utilisant HAVING.....                             | 162        |
| Grouper en utilisant Group Concat.....                        | 162        |
| GROUP BY avec les fonctions AGGREGATE.....                    | 163        |
| <b>Chapitre 47: Partitionnement.....</b>                      | <b>166</b> |
| Remarques.....  | 166        |
| Exemples.....   | 166        |
| Partitionnement RANGE.....                                    | 166        |
| Partitionnement LIST.....                                     | 167        |
| Partitionnement HASH.....                                     | 168        |
| <b>Chapitre 48: Personnaliser PS1.....</b>                    | <b>169</b> |
| Exemples.....   | 169        |
| Personnaliser MySQL PS1 avec la base de données actuelle..... | 169        |
| PS1 personnalisé via le fichier de configuration MySQL.....   | 169        |
| <b>Chapitre 49: Pointes.....</b>                              | <b>170</b> |
| Exemples.....   | 170        |
| Utilisation de backticks.....                                 | 170        |
| <b>Chapitre 50: PREPARE Déclarations.....</b>                 | <b>172</b> |
| Syntaxe.....  | 172        |
| Exemples.....   | 172        |
| PREPARE, EXECUTE et DEALLOCATE PREPARE Déclarations.....      | 172        |
| Construire et exécuter.....                                   | 172        |
| Alter table avec add column.....                              | 173        |
| <b>Chapitre 51: Recherche en texte intégral.....</b>          | <b>174</b> |
| Introduction.....   | 174        |
| Remarques.....  | 174        |

|   |            |
|---|------------|
| Exemples.....   | 174        |
| Recherche FULLTEXT simple.....  | 174        |
| Recherche BOOLEAN simple.....   | 174        |
| Recherche FULLTEXT sur plusieurs colonnes.....  | 175        |
| <b>Chapitre 52: Récupérer du mot de passe root perdu.....</b>                                       | <b>176</b> |
| Exemples.....   | 176        |
| Définir le mot de passe root, activer l'utilisateur root pour l'accès socket et http.....           | 176        |
| <b>Chapitre 53: Récupérer et réinitialiser le mot de passe root par défaut pour MySQL 5.7+.....</b> | <b>177</b> |
| Introduction.....   | 177        |
| Remarques.....  | 177        |
| Exemples.....   | 177        |
| Que se passe-t-il lorsque le démarrage initial du serveur.....                                      | 177        |
| Comment changer le mot de passe root en utilisant le mot de passe par défaut.....                   | 177        |
| réinitialiser le mot de passe root lorsque "/ var / run / mysqld" pour le fichier de socke.....     | 178        |
| <b>Chapitre 54: Réplication.....</b>  | <b>180</b> |
| Remarques.....  | 180        |
| Exemples.....   | 180        |
| Maître - Configuration de la réplication esclave.....   | 180        |
| Erreurs de réplication.....   | 183        |
| <b>Chapitre 55: Requêtes pivot.....</b>   | <b>185</b> |
| Remarques.....  | 185        |
| Exemples.....   | 185        |
| Création d'une requête pivot.....   | 185        |
| <b>Chapitre 56: Routines stockées (procédures et fonctions).....</b>                                | <b>187</b> |
| Paramètres.....   | 187        |
| Remarques.....  | 187        |
| Exemples.....   | 187        |
| Créer une fonction.....   | 187        |
| Créer une procédure avec une préparation construite.....  | 188        |
| Procédure stockée avec les paramètres IN, OUT, INOUT.....   | 189        |
| Des curseurs.....   | 190        |
| Plusieurs ResultSets.....   | 192        |

|   |            |
|---|------------|
| Créer une fonction.....   | 192        |
| <b>Chapitre 57: Sauvegarde avec mysqldump.....</b>                                  | <b>193</b> |
| Syntaxe.....  | 193        |
| Paramètres.....   | 193        |
| Remarques.....  | 194        |
| Exemples.....   | 194        |
| Création d'une sauvegarde d'une base de données ou d'une table.....                 | 194        |
| Spécifier le nom d'utilisateur et le mot de passe.....                              | 195        |
| Restauration d'une sauvegarde d'une base de données ou d'une table.....             | 195        |
| mysqldump depuis un serveur distant avec compression.....                           | 196        |
| restaurer un fichier mysqldump sans compresser.....                                 | 196        |
| Sauvegarde directe sur Amazon S3 avec compression.....                              | 196        |
| Transfert de données d'un serveur MySQL vers un autre.....                          | 196        |
| Sauvegarde de la base de données avec des procédures stockées et des fonctions..... | 197        |
| <b>Chapitre 58: Sécurité via GRANTS.....</b>  | <b>198</b> |
| Exemples.....   | 198        |
| Meilleur entraînement.....  | 198        |
| Hôte (de l'utilisateur @ hôte).....   | 198        |
| <b>Chapitre 59: SÉLECTIONNER.....</b>   | <b>200</b> |
| Introduction.....   | 200        |
| Syntaxe.....  | 200        |
| Remarques.....  | 200        |
| Exemples.....   | 200        |
| SELECT par nom de colonne.....  | 200        |
| SÉLECTIONNEZ toutes les colonnes (*).....   | 201        |
| SELECT avec WHERE.....  | 202        |
| <b>Requête avec un SELECT imbriqué dans la clause WHERE.....</b>                    | <b>202</b> |
| SELECT avec LIKE (%).....   | 202        |
| SELECT avec Alias (AS).....   | 204        |
| SELECT avec une clause LIMIT.....   | 204        |
| SELECT avec DISTINCT.....   | 205        |
| SELECT avec LIKE (____).....  | 206        |



|   |            |
|---|------------|
| SELECT avec CASE ou IF.....   | 206        |
| SELECT avec entre.....  | 207        |
| SELECT avec plage de dates.....   | 208        |
| <b>Chapitre 60: SYNDICAT.....</b>   | <b>209</b> |
| Syntaxe.....  | 209        |
| Remarques.....  | 209        |
| Exemples.....   | 209        |
| Combiner les instructions SELECT avec UNION.....  | 209        |
| COMMANDÉ PAR.....   | 209        |
| Pagination via OFFSET.....  | 210        |
| Combinaison de données avec différentes colonnes.....   | 210        |
| UNION ALL et UNION.....   | 210        |
| Combinaison et fusion de données sur différentes tables MySQL avec les mêmes colonnes en l..... | 211        |
| <b>Chapitre 61: Table de chute.....</b>   | <b>212</b> |
| Syntaxe.....  | 212        |
| Paramètres.....   | 212        |
| Exemples.....   | 212        |
| Table de chute.....   | 212        |
| Déposer des tables de la base de données.....   | 213        |
| <b>Chapitre 62: Table de mappage plusieurs-à-plusieurs.....</b>                                 | <b>214</b> |
| Remarques.....  | 214        |
| Exemples.....   | 214        |
| Schéma typique.....   | 214        |
| <b>Chapitre 63: Tableau dynamique de pivotement à l'aide de l'instruction préparée.....</b>     | <b>215</b> |
| Exemples.....   | 215        |
| Dé-pivoter un ensemble dynamique de colonnes en fonction de la condition.....                   | 215        |
| <b>Chapitre 64: Tables temporaires.....</b>   | <b>218</b> |
| Exemples.....   | 218        |
| Créer une table temporaire.....   | 218        |
| Drop Table temporaire.....  | 218        |
| <b>Chapitre 65: Temps avec précision de la seconde.....</b>                                     | <b>220</b> |

|   |            |
|---|------------|
| Remarques.....  | 220        |
| Exemples.....   | 220        |
| Obtenez l'heure actuelle avec une précision en milliseconde.....                      | 220        |
| Obtenez l'heure actuelle sous une forme qui ressemble à un horodatage Javascript..... | 220        |
| Créez un tableau avec des colonnes pour stocker le temps inférieur à la seconde.....  | 221        |
| Convertissez une valeur de date / heure en millisecondes de précision en texte.....   | 221        |
| Stocker un horodatage Javascript dans une colonne TIMESTAMP.....                      | 221        |
| <b>Chapitre 66: Traiter des données rares ou manquantes.....</b>                      | <b>223</b> |
| Exemples.....   | 223        |
| Utilisation de colonnes contenant des valeurs NULL.....                               | 223        |
| <b>Chapitre 67: Transaction.....</b>  | <b>226</b> |
| Exemples.....   | 226        |
| Commencer la transaction.....   | 226        |
| COMMIT, ROLLBACK et AUTOCOMMIT.....   | 227        |
| Transaction à l'aide du pilote JDBC.....  | 230        |
| <b>Chapitre 68: TRIGGERS.....</b>   | <b>233</b> |
| Syntaxe.....  | 233        |
| Remarques.....  | 233        |
| <b>POUR CHAQUE RANG.....</b>  | <b>233</b> |
| <b>Créer ou remplacer un déclencheur.....</b>   | <b>233</b> |
| Exemples.....   | 234        |
| Déclencheur de base.....  | 234        |
| Types de déclencheurs.....  | 234        |
| <b>Timing.....</b>  | <b>234</b> |
| <b>Événement déclencheur.....</b>   | <b>235</b> |
| <b>Avant Insérer exemple de déclenchement.....</b>                                    | <b>235</b> |
| <b>Avant mise à jour exemple de déclenchement.....</b>                                | <b>235</b> |
| <b>Après Supprimer exemple de déclenchement.....</b>                                  | <b>235</b> |
| <b>Chapitre 69: Types de données.....</b>   | <b>237</b> |
| Exemples.....   | 237        |
| Moulage implicite / automatique.....  | 237        |

|   |            |
|---|------------|
| VARCHAR (255) - ou non.....   | 237        |
| INT comme AUTO_INCREMENT.....   | 238        |
| Autres.....   | 238        |
| Introduction (numérique).....   | 239        |
| Types entiers.....  | 239        |
| Types de points fixes.....  | 240        |
| Décimal.....  | 240        |
| Types de virgule flottante.....   | 240        |
| Type de valeur de bit.....  | 241        |
| CHAR (n).....   | 241        |
| DATE, DATETIME, TIMESTAMP, YEAR et TIME.....                                    | 241        |
| <b>Chapitre 70: Un à plusieurs.....</b>   | <b>243</b> |
| Introduction.....   | 243        |
| Remarques.....  | 243        |
| Exemples.....   | 243        |
| Exemples de tables d'entreprise.....  | 243        |
| Gérer les employés par un seul responsable.....                                 | 244        |
| Obtenir le gestionnaire pour un seul employé.....                               | 244        |
| <b>Chapitre 71: Unions MySQL.....</b>   | <b>245</b> |
| Syntaxe.....  | 245        |
| Remarques.....  | 245        |
| Exemples.....   | 245        |
| Opérateur syndical.....   | 245        |
| Union TOUT.....   | 246        |
| UNION ALL With WHERE.....   | 246        |
| <b>Chapitre 72: Utiliser des variables.....</b>                                 | <b>248</b> |
| Exemples.....   | 248        |
| Définition des variables.....   | 248        |
| Numéro de ligne et groupe en utilisant des variables dans Select Statement..... | 249        |
| <b>Chapitre 73: VUE.....</b>  | <b>251</b> |
| Syntaxe.....  | 251        |
| Paramètres.....   | 251        |

|  |            |
|--|------------|
| Remarques.....                           | 251        |
| Exemples.....                            | 252        |
| Créer une vue.....                       | 252        |
| Une vue de deux tables.....              | 253        |
| Mise à jour d'une table via une vue..... | 253        |
| DROPPING A VIEW.....                     | 254        |
| <b>Crédits.....</b>                      | <b>255</b> |

# À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [mysql](#)

It is an unofficial and free MySQL ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official MySQL.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

---

# Chapitre 1: Démarrer avec MySQL

## Remarques



[MySQL](#) est un système de gestion de base de données relationnelle (SGBDR) open source développé et pris en charge par Oracle Corporation.

MySQL est [pris en charge](#) sur un grand nombre de plates-formes, y compris les variantes Linux, OS X et Windows. Il possède également des [API](#) pour un grand nombre de langages, notamment C, C ++, Java, Lua, .Net, Perl, PHP, Python et Ruby.

[MariaDB](#) est un fork de MySQL avec un [ensemble de fonctionnalités légèrement différent](#) . Il est entièrement compatible avec MySQL pour la plupart des applications.

## Versions

| Version | Date de sortie |
|---------|----------------|
| 1.0     | 1995-05-23     |
| 3.19    | 1996-12-01     |
| 3.20    | 1997-01-01     |
| 3.21    | 1998-10-01     |
| 3.22    | 1999-10-01     |
| 3.23    | 2001-01-22     |
| 4.0     | 2003-03-01     |
| 4.1     | 2004-10-01     |
| 5.0     | 2005-10-01     |
| 5.1     | 2008-11-27     |
| 5.5     | 2010-11-01     |
| 5.6     | 2013-02-01     |

| Version | Date de sortie |
|---------|----------------|
| 5.7     | 2015-10-01     |

## Exemples

### Commencer

#### Créer une base de données dans MySQL

```
CREATE DATABASE mydb;
```

Valeur de retour:

Requête OK, 1 ligne affectée (0.05 sec)

---

#### Utiliser la base de données créée `mydb`

```
USE mydb;
```

Valeur de retour:

Base de données modifiée

#### Créer une table dans MySQL

```
CREATE TABLE mytable  
(  
  id          int unsigned NOT NULL auto_increment,  
  username    varchar(100) NOT NULL,  
  email       varchar(100) NOT NULL,  
  PRIMARY KEY (id)  
);
```

`CREATE TABLE mytable` va créer une nouvelle table appelée `mytable`.

`id int unsigned NOT NULL auto_increment` crée la colonne `id`, ce type de champ assignera un identifiant numérique unique à chaque enregistrement de la table (ce qui signifie que deux lignes ne peuvent pas avoir le même `id` dans ce cas), MySQL affectera automatiquement une nouvelle valeur unique dans le champ `id` l'enregistrement (commençant par 1).

Valeur de retour:

Requête OK, 0 lignes affectées (0.10 sec)

---

#### Insérer une ligne dans une table MySQL

```
INSERT INTO mytable ( username, email )
```

```
VALUES ( "myuser", "myuser@example.com" );
```

Exemple de valeur de retour:

Requête OK, 1 ligne affectée (0.06 sec)

Les `strings varchar` aka peuvent également être insérées en utilisant des guillemets simples:

```
INSERT INTO mytable ( username, email )  
VALUES ( 'username', 'username@example.com' );
```

---

## Mise à jour d'une ligne dans une table MySQL

```
UPDATE mytable SET username="myuser" WHERE id=8
```

Exemple de valeur de retour:

Requête OK, 1 ligne affectée (0.06 sec)

La valeur `int` peut être insérée dans une requête sans guillemets. Les chaînes et les dates doivent être entre guillemets simples ' ou des guillemets doubles " .

---

## Supprimer une ligne dans une table MySQL

```
DELETE FROM mytable WHERE id=8
```

Exemple de valeur de retour:

Requête OK, 1 ligne affectée (0.06 sec)

Cela supprimera la ligne ayant l' `id` est 8.

---

## Sélection de lignes en fonction des conditions dans MySQL

```
SELECT * FROM mytable WHERE username = "myuser";
```

Valeur de retour:

```
+----+-----+-----+  
| id | username | email |  
+----+-----+-----+  
| 1 | myuser | myuser@example.com |  
+----+-----+-----+
```

1 rangée dans l'ensemble (0.00 sec)



## Afficher la liste des bases de données existantes

```
SHOW databases;
```

Valeur de retour:

```
+-----+
| Databases |
+-----+
| information_schema |
| mydb |
+-----+
```

2 lignes dans l'ensemble (0.00 sec)

Vous pouvez penser à "information\_schema" comme une "base de données principale" qui donne accès aux métadonnées de la base de données.

---

## Afficher les tables dans une base de données existante

```
SHOW tables;
```

Valeur de retour:

```
+-----+
| Tables_in_mydb |
+-----+
| mytable |
+-----+
```

1 rangée dans l'ensemble (0.00 sec)

---

## Afficher tous les champs d'une table

```
DESCRIBE databaseName.tableName;
```

ou, si vous utilisez déjà une base de données:

```
DESCRIBE tableName;
```

Valeur de retour:

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null  | Key    | Default          | Extra |
+-----+-----+-----+-----+-----+-----+
| fieldname  | fieldvaluetype | NO/YES | keytype | defaultfieldvalue |      |
+-----+-----+-----+-----+-----+-----+
```

Extra peut contenir `auto_increment` par exemple.

`key` fait référence au type de clé pouvant affecter le champ. Primaire (PRI), Unique (UNI) ...

n rangée dans l'ensemble (0.00 sec)

Où n est le nombre de champs de la table.

---

## Créer un utilisateur

Tout d'abord, vous devez créer un utilisateur, puis attribuer à l'utilisateur des autorisations sur certaines bases de données / tables. Lors de la création de l'utilisateur, vous devez également spécifier d'où cet utilisateur peut se connecter.

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

Permet de créer un utilisateur qui ne peut se connecter qu'à l'ordinateur local sur lequel la base de données est hébergée.

```
CREATE USER 'user'@'%' IDENTIFIED BY 'some_password';
```

Permet de créer un utilisateur pouvant se connecter de n'importe où (à l'exception de la machine locale).

Exemple de valeur de retour:

Requête OK, 0 lignes affectées (0.00 sec)

## Ajouter des privilèges

Accordez à l'utilisateur des privilèges de base communs pour toutes les tables de la base de données spécifiée:

```
GRANT SELECT, INSERT, UPDATE ON databaseName.* TO 'userName'@'localhost';
```

Accordez tous les privilèges à l'utilisateur pour toutes les tables de toutes les bases de données (attention à ceci):

```
GRANT ALL ON *.* TO 'userName'@'localhost' WITH GRANT OPTION;
```

Comme démontré ci-dessus, `*.*` Cible toutes les bases de données et tables, `databaseName.*` Cible toutes les tables de la base de données spécifique. Il est également possible de spécifier la base de données et la table comme si vous aviez un nom de `databaseName.tableName` .

`WITH GRANT OPTION` doit être omis si l'utilisateur n'a pas besoin d'accorder des privilèges aux autres utilisateurs.

Les privilèges peuvent être **soit**

```
ALL
```

**ou** une combinaison des éléments suivants, chacun séparé par une virgule (liste non exhaustive).

```
SELECT
INSERT
UPDATE
DELETE
CREATE
DROP
```

## Remarque

En règle générale, vous devez éviter d'utiliser des noms de colonnes ou de tables contenant des espaces ou des mots réservés en SQL. Par exemple, il est préférable d'éviter les noms tels que `table OU first name`.

Si vous devez utiliser de tels noms, placez-les entre des délimiteurs `` back-tick ``. Par exemple:

```
CREATE TABLE `table`
(
  `first name` VARCHAR(30)
);
```

Une requête contenant les délimiteurs back-tick sur cette table pourrait être:

```
SELECT `first name` FROM `table` WHERE `first name` LIKE 'a%';
```

## Exemples de schémas d'information

# Liste des processus

Cela affichera toutes les requêtes actives et en sommeil dans cet ordre, puis de combien de temps.

```
SELECT * FROM information_schema.PROCESSLIST ORDER BY INFO DESC, TIME DESC;
```

Ceci est un peu plus en détail sur les délais comme il est en secondes par défaut

```
SELECT ID, USER, HOST, DB, COMMAND,
TIME as time_seconds,
ROUND(TIME / 60, 2) as time_minutes,
ROUND(TIME / 60 / 60, 2) as time_hours,
STATE, INFO
FROM information_schema.PROCESSLIST ORDER BY INFO DESC, TIME DESC;
```

# Procédure stockée Recherche

Recherchez facilement des mots et des caractères génériques dans toutes les `Stored Procedures`.

```
SELECT * FROM information_schema.ROUTINES WHERE ROUTINE_DEFINITION LIKE '%word%';
```

Lire Démarrer avec MySQL en ligne: <https://riptutorial.com/fr/mysql/topic/302/demarrer-avec-mysql>

# Chapitre 2: ALTER TABLE

## Syntaxe

- ALTER [IGNORE] TABLE **nom\_de\_table** [ **alter\_specification** [, alter\_specification] ...] [partition\_options]

## Remarques

```
alter_specification: table_options
| ADD [COLUMN] col_name column_definition [FIRST | AFTER col_name ]
| ADD [COLUMN] (col_name column_definition,...)
| ADD {INDEX|KEY} [index_name] [index_type] (index_col_name,...) [index_option] ...
| ADD [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...) [index_option]
...
| ADD [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [index_name] [index_type]
(index_col_name,...) [index_option] ...
| ADD FULLTEXT [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
| ADD SPATIAL [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
| ADD [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (index_col_name,...)
reference_definition
| ALGORITHM [=] {DEFAULT|INPLACE|COPY}
| ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
| CHANGE [COLUMN] old_col_name new_col_name column_definition [FIRST|AFTER col_name]
| LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
| MODIFY [COLUMN] col_name column_definition [FIRST | AFTER col_name]
| DROP [COLUMN] col_name
| DROP PRIMARY KEY
| DROP {INDEX|KEY} index_name
| DROP FOREIGN KEY fk_symbol
| DISABLE KEYS
| ENABLE KEYS
| RENAME [TO|AS] new_tbl_name
| RENAME {INDEX|KEY} old_index_name TO new_index_name
| ORDER BY col_name [, col_name] ...
| CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
| [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
| DISCARD TABLESPACE
| IMPORT TABLESPACE
| FORCE
| {WITHOUT|WITH} VALIDATION
| ADD PARTITION (partition_definition)
| DROP PARTITION partition_names
| DISCARD PARTITION {partition_names | ALL} TABLESPACE
| IMPORT PARTITION {partition_names | ALL} TABLESPACE
| TRUNCATE PARTITION {partition_names | ALL}
| COALESCE PARTITION number
| REORGANIZE PARTITION partition_names INTO (partition_definitions)
| EXCHANGE PARTITION partition_name WITH TABLE tbl_name [{WITH|WITHOUT} VALIDATION]
| ANALYZE PARTITION {partition_names | ALL}
| CHECK PARTITION {partition_names | ALL}
| OPTIMIZE PARTITION {partition_names | ALL}
| REBUILD PARTITION {partition_names | ALL}
| REPAIR PARTITION {partition_names | ALL}
| REMOVE PARTITIONING
```

```
| UPGRADE PARTITIONING
index_col_name: col_name [(length)] [ASC | DESC]
index_type: USING {BTREE | HASH}
index_option: KEY_BLOCK_SIZE [=] value
| index_type
| WITH PARSEr parser_name
| COMMENT 'string'
```

table\_options: table\_option [[,] table\_option] ... (see options) [CREATE TABLE](#) options)

partition\_options: (see options) [CREATE TABLE](#) options)

Réf: [MySQL 5.7 Reference Manual / ... / Syntaxe ALTER TABLE / 14.1.8 Syntaxe ALTER TABLE](#)

## Exemples

### Changement de moteur de stockage; reconstruire la table; changez file\_per\_table

Par exemple, si `t1` n'est actuellement pas une table InnoDB, cette instruction modifie son moteur de stockage en InnoDB:

```
ALTER TABLE t1 ENGINE = InnoDB;
```

Si la table est déjà InnoDB, cela reconstruira la table et ses index et aura un effet similaire à `OPTIMIZE TABLE`. Vous pouvez obtenir une amélioration de l'espace disque.

Si la valeur de `innodb_file_per_table` est actuellement différente de celle utilisée lors de la construction de `t1`, cela convertira en (ou depuis) `file_per_table`.

## ALTER COLONNE DE TABLE

```
CREATE DATABASE stackoverflow;

USE stackoverflow;

Create table stack(
  id_user int NOT NULL,
  username varchar(30) NOT NULL,
  password varchar(30) NOT NULL
);

ALTER TABLE stack ADD COLUMN submit date NOT NULL; -- add new column
ALTER TABLE stack DROP COLUMN submit; -- drop column
ALTER TABLE stack MODIFY submit DATETIME NOT NULL; -- modify type column
ALTER TABLE stack CHANGE submit submit_date DATETIME NOT NULL; -- change type and name of column
ALTER TABLE stack ADD COLUMN mod_id INT NOT NULL AFTER id_user; -- add new column after existing column
```

## ALTER table ajouter INDEX

Pour améliorer les performances, vous pouvez ajouter des index aux colonnes.

```
ALTER TABLE TABLE_NAME ADD INDEX `index_name` (`column_name`)
```

modification pour ajouter des index composites (plusieurs colonnes)

```
ALTER TABLE TABLE_NAME ADD INDEX `index_name` (`col1`,`col2`)
```

## Modifier la valeur d'incrément automatique

La modification d'une valeur d'incrément automatique est utile lorsque vous ne souhaitez pas créer d'espace dans une colonne AUTO\_INCREMENT après une suppression massive.

Par exemple, vous avez beaucoup de lignes indésirables (publicité) affichées dans votre table, vous les avez supprimées et vous souhaitez corriger l'écart en valeurs d'incrément automatique. Supposons que la valeur MAX de la colonne AUTO\_INCREMENT est de 100 maintenant. Vous pouvez utiliser les éléments suivants pour corriger la valeur d'incrément automatique.

```
ALTER TABLE your_table_name AUTO_INCREMENT = 101;
```

## Modification du type d'une colonne de clé primaire

```
ALTER TABLE fish_data.fish DROP PRIMARY KEY;  
ALTER TABLE fish_data.fish MODIFY COLUMN fish_id DECIMAL(20,0) NOT NULL PRIMARY KEY;
```

Une tentative de modification du type de cette colonne sans supprimer d'abord la clé primaire entraînerait une erreur.

## Changer la définition de la colonne

La modification de la définition d'une colonne de base de données, la requête ci-dessous peut être utilisée par exemple, si nous avons ce schéma de base de données

```
users (  
  firstname varchar(20),  
  lastname varchar(20),  
  age char(2)  
)
```

Pour changer le type de colonne `age` de `char` en `int`, nous utilisons la requête ci-dessous:

```
ALTER TABLE users CHANGE age age tinyint UNSIGNED NOT NULL;
```

Le format général est:

```
ALTER TABLE table_name CHANGE column_name new_column_definition
```

## Renommer une base de données MySQL

Il n'y a pas de commande unique pour renommer une base de données MySQL, mais une solution simple peut être utilisée pour y parvenir en sauvegardant et en restaurant:

```
mysqladmin -uroot -p<password> create <new name>
mysqldump -uroot -p<password> --routines <old name> | mysql -uroot -pmypassword <new name>
mysqladmin -uroot -p<password> drop <old name>
```

### Pas:

1. Copiez les lignes ci-dessus dans un éditeur de texte.
2. Remplacez toutes les références à `<old name>`, `<new name>` et `<password>` (+ éventuellement `root` pour utiliser un autre utilisateur) avec les valeurs appropriées.
3. Exécutez un par un sur la ligne de commande (en supposant que le dossier "bin" de MySQL se trouve dans le chemin et que vous entrez "y" lorsque vous y êtes invité).

### Étapes alternatives:

Renommez (déplacez) chaque table d'un db à l'autre. Faites ceci pour chaque table:

```
RENAME TABLE `<old db>`.`<name>` TO `<new db>`.`<name>`;
```

Vous pouvez créer ces déclarations en faisant quelque chose comme

```
SELECT CONCAT('RENAME TABLE old_db.', table_name, ' TO ',
              'new_db.', table_name)
FROM information_schema.TABLES
WHERE table_schema = 'old_db';
```

Attention. N'essayez pas de créer une table ou une base de données en déplaçant simplement des fichiers sur le système de fichiers. Cela a bien fonctionné dans le bon vieux temps de MyISAM, mais dans les nouveaux jours d'InnoDB et des tablespaces, cela ne fonctionnera pas. Surtout lorsque le "Dictionnaire de données" est déplacé du système de fichiers dans les tables InnoDB du système, probablement dans la prochaine version majeure. Déplacer (par opposition à simplement `DROPPing`) une `PARTITION` d'une table InnoDB nécessite l'utilisation de "tablespaces transportables". Dans un avenir proche, il n'y aura même pas de fichier à atteindre.

## Echanger les noms de deux bases de données MySQL

Les commandes suivantes peuvent être utilisées pour permuter les noms de deux bases de données MySQL (`<db1>` et `<db2>`):

```
mysqladmin -uroot -p<password> create swaptemp
mysqldump -uroot -p<password> --routines <db1> | mysql -uroot -p<password> swaptemp
mysqladmin -uroot -p<password> drop <db1>
mysqladmin -uroot -p<password> create <db1>
mysqldump -uroot -p<password> --routines <db2> | mysql -uroot -p<password> <db1>
mysqladmin -uroot -p<password> drop <db2>
mysqladmin -uroot -p<password> create <db2>
```



```
mysqldump -uroot -p<password> --routines swaptemp | mysql -uroot -p<password> <db2>
mysqladmin -uroot -p<password> drop swaptemp
```

### Pas:

1. Copiez les lignes ci-dessus dans un éditeur de texte.
2. Remplacez toutes les références à `<db1>` , `<db2>` et `<password>` (+ éventuellement `root` pour utiliser un autre utilisateur) avec les valeurs appropriées.
3. Exécutez un par un sur la ligne de commande (en supposant que le dossier "bin" de MySQL se trouve dans le chemin et que vous entrez "y" lorsque vous y êtes invité).

## Renommer une table MySQL

Renommer une table peut être fait en une seule commande:

```
RENAME TABLE `<old name>` TO `<new name>`;
```

La syntaxe suivante fait exactement la même chose:

```
ALTER TABLE `<old name>` RENAME TO `<new name>`;
```

Si vous renommez une table temporaire, vous devez utiliser la version `ALTER TABLE` de la syntaxe.

### Pas:

1. Remplacez `<old name>` et `<new name>` dans la ligne ci-dessus par les valeurs correspondantes. *Remarque: Si la table est déplacée vers une autre base de données, le `dbname . tablename` syntaxe de `tablename` peut être utilisée pour `<old name>` et / ou `<new name>` .*
2. Exécutez-le sur la base de données appropriée sur la ligne de commande MySQL ou sur un client tel que MySQL Workbench. *Remarque: L'utilisateur doit disposer des privilèges `ALTER` et `DROP` sur l'ancienne table et `CREATE` et `INSERT` sur la nouvelle.*

## Renommer une colonne dans une table MySQL

Renommer une colonne peut être fait dans une seule instruction, mais en plus du nouveau nom, la "définition de colonne" (c'est-à-dire son type de données et d'autres propriétés facultatives telles que nullité, incrémentation automatique, etc.) doit également être spécifiée.

```
ALTER TABLE `<table name>` CHANGE `<old name>` `<new name>` <column definition>;
```

### Pas:

1. Ouvrez la ligne de commande MySQL ou un client tel que MySQL Workbench.
2. Exécutez l'instruction suivante: `SHOW CREATE TABLE <table name>;` (en remplaçant `<table name>` par la valeur correspondante).
3. Notez la définition complète de la colonne à renommer (*c'est-à-dire tout ce qui apparaît après le nom de la colonne mais avant la virgule le séparant du nom de colonne suivant*) .
4. Remplacez `<old name>` , `<new name>` et `<column definition>` dans la ligne ci-dessus par les

valeurs appropriées, puis exécutez-la.

Lire ALTER TABLE en ligne: <https://riptutorial.com/fr/mysql/topic/2627/alter-table>

# Chapitre 3: Arithmétique

## Remarques

MySQL, sur la plupart des machines, utilise l' [arithmétique flottante IEEE 754 64 bits](#) pour ses calculs.

Dans les contextes entiers, il utilise l'arithmétique entière.

- `RAND()` n'est pas un générateur de nombres aléatoires parfait. Il est principalement utilisé pour générer rapidement des nombres pseudo-aléatoires

## Exemples

### Opérateurs arithmétiques

MySQL fournit les opérateurs arithmétiques suivants

| Opérateur | prénom           | Exemple   |
|-----------|------------------|---|
| +         | Une addition     | <code>SELECT 3+5; -&gt; 8</code><br><code>SELECT 3.5+2.5; -&gt; 6.0</code><br><code>SELECT 3.5+2; -&gt; 5.5</code>  |
| -         | Soustraction     | <code>SELECT 3-5; -&gt; -2</code>   |
| *         | Multiplication   | <code>SELECT 3 * 5; -&gt; 15</code>   |
| /         | Division         | <code>SELECT 20 / 4; -&gt; 5</code><br><code>SELECT 355 / 113; -&gt; 3.1416</code><br><code>SELECT 10.0 / 0; -&gt; NULL</code>  |
| DIV       | Division entière | <code>SELECT 5 DIV 2; -&gt; 2</code>  |
| % ou MOD  | Modulo           | <code>SELECT 7 % 3; -&gt; 1</code><br><code>SELECT 15 MOD 4 -&gt; 3</code><br><code>SELECT 15 MOD -4 -&gt; 3</code><br><code>SELECT -15 MOD 4 -&gt; -3</code><br><code>SELECT -15 MOD -4 -&gt; -3</code><br><code>SELECT 3 MOD 2.5 -&gt; 0.5</code> |

## BIGINT

Si les nombres de votre arithmétique sont tous des nombres entiers, MySQL utilise le type de données entier `BIGINT` (signé 64 bits) pour faire son travail. Par exemple:

```
select (1024 * 1024 * 1024 * 1024 *1024 * 1024) + 1 -> 1,152,921,504,606,846,977
```

et

```
select (1024 * 1024 * 1024 * 1024 *1024 * 1024 * 1024 -> BIGINT hors plage
```

## DOUBLE

Si des nombres dans votre arithmétique sont fractionnaires, MySQL utilise l' [arithmétique flottante IEEE 754 64 bits](#) . Vous devez faire attention lorsque vous utilisez l'arithmétique à virgule flottante, car de nombreux [nombres à virgule flottante sont, par nature, des approximations plutôt que des valeurs exactes](#) .

## Constantes Mathématiques

### Pi

Le texte suivant renvoie la valeur de `PI` formatée à 6 décimales. La valeur réelle est bonne pour `DOUBLE` ;

```
SELECT PI (); -> 3.141593
```

### Trigonométrie (SIN, COS)

Les angles sont en radians, pas en degrés. Tous les calculs sont effectués en [virgule flottante IEEE 754 64 bits](#) . Tous les calculs en virgule flottante sont sujets à de petites erreurs, appelées erreurs  $\epsilon$  ([epsilon](#)) , évitez donc de les comparer pour obtenir une égalité. Il n'y a aucun moyen d'éviter ces erreurs lors de l'utilisation de virgule flottante; ils sont intégrés à la technologie.

Si vous utilisez les valeurs `DECIMAL` dans les calculs trigonométriques, elles sont implicitement converties en virgule flottante, puis de nouveau en décimales.

### Sinus

Retourne le sinus d'un nombre X exprimé en radians

```
SELECT SIN(PI()); -> 1.2246063538224e-16
```

### Cosinus

Retourne le cosinus de X quand X est donné en radians

```
SELECT COS(PI()); -> -1
```

# Tangente

Renvoie la tangente d'un nombre X exprimé en radians. Notez que le résultat est très proche de zéro, mais pas exactement zéro. Ceci est un exemple de machine  $\epsilon$ .

```
SELECT TAN(PI()); -> -1.2246063538224e-16
```

# Arc Cosinus (cosinus inverse)

Retourne l'arc cosinus de X si X est compris entre -1 to 1

```
SELECT ACOS(1); -> 0
SELECT ACOS(1.01); -> NULL
```

# Arc Sine (sinus inverse)

Retourne l'arc sinus de X si X est compris entre -1 to 1

```
SELECT ASIN(0.2); -> 0.20135792079033
```

# Arc tangente (tangente inverse)

ATAN(x) renvoie l'arc tangente d'un nombre unique.

```
SELECT ATAN(2); -> 1.1071487177941
```

ATAN2(X, Y) renvoie l'arc tangent des deux variables X et Y. Il est similaire au calcul de l'arc tangent de Y / X. Mais il est numériquement plus robuste: t fonctionne correctement lorsque X est proche de zéro et que les signes des deux arguments sont utilisés pour déterminer le quadrant du résultat.

Les meilleures pratiques suggèrent d'écrire des formules pour utiliser ATAN2() plutôt ATAN() mesure du possible.

```
ATAN2(1,1); -> 0.7853981633974483 (45 degrees)
ATAN2(1,-1); -> 2.356194490192345 (135 degrees)
ATAN2(0, -1); -> PI (180 degrees) don't try ATAN(-1 / 0)... it won't work
```

# Cotangente

Renvoie la cotangente de X

```
SELECT COT(12); -> -1.5726734063977
```

# Conversion

```
SELECT RADIANS(90) -> 1.5707963267948966
SELECT SIN(RADIANS(90)) -> 1
SELECT DEGREES(1), DEGREES(PI()) -> 57.29577951308232, 180
```

## Arrondi (ROUND, FLOOR, CEIL)

### Arrondir un nombre décimal à une valeur entière

Pour les valeurs numériques exactes (par exemple, `DECIMAL`): Si la première décimale d'un nombre est supérieure ou égale à 5, cette fonction arrondira le nombre à l'entier suivant, à *partir de zéro*. Si cette décimale est 4 ou moins, cette fonction arrondira à la prochaine valeur entière la *plus proche de zéro*.

```
SELECT ROUND(4.51) -> 5
SELECT ROUND(4.49) -> 4
SELECT ROUND(-4.51) -> -5
```

Pour les valeurs numériques approximatives (par exemple `DOUBLE`): Le résultat de la fonction `ROUND()` dépend de la bibliothèque C; sur de nombreux systèmes, cela signifie que `ROUND()` utilise le *tour à la règle la plus proche*:

```
SELECT ROUND(45e-1) -> 4 -- The nearest even value is 4
SELECT ROUND(55e-1) -> 6 -- The nearest even value is 6
```

### Arrondir un nombre

Pour arrondir une utilisation du numéro soit la `CEIL()` ou `CEILING()` fonction

```
SELECT CEIL(1.23) -> 2
SELECT CEILING(4.83) -> 5
```

### Arrondissez un nombre

Pour arrondir un nombre, utilisez la fonction `FLOOR()`

```
SELECT FLOOR(1.99) -> 1
```

`FLOOR` et `CEIL` vont vers / loin de l'infini:

```
SELECT FLOOR(-1.01), CEIL(-1.01) -> -2 and -1
SELECT FLOOR(-1.99), CEIL(-1.99) -> -2 and -1
```

# Arrondissez un nombre décimal à un nombre spécifié de décimales.

```
SELECT ROUND(1234.987, 2) -> 1234.99
SELECT ROUND(1234.987, -2) -> 1200
```

La discussion de haut contre bas et "5" s'applique aussi.

## Augmenter un nombre à une puissance (POW)

Pour élever un nombre  $x$  à une puissance  $y$ , utilisez les fonctions `POW()` ou `POWER()`

```
SELECT POW(2,2); => 4
SELECT POW(4,2); => 16
```

## Racine Carrée (SQRT)

Utilisez la fonction `SQRT()`. Si le nombre est négatif, `NULL` sera renvoyé

```
SELECT SQRT(16); -> 4
SELECT SQRT(-3); -> NULL
```

## Nombres Aléatoires (RAND)

### Générer un nombre aléatoire

Pour générer un nombre à virgule flottante pseudo-aléatoire compris entre 0 et 1, utilisez la fonction `RAND()`

Supposons que vous ayez la requête suivante

```
SELECT i, RAND() FROM t;
```

Cela retournera quelque chose comme ça

| je | RAND()           |
|----|------------------|
| 1  | 0.6191438870682  |
| 2  | 0.93845168309142 |
| 3  | 0.83482678498591 |

### Nombre aléatoire dans une plage

Pour générer un nombre aléatoire dans la plage  $a \leq n \leq b$ , vous pouvez utiliser la formule suivante

```
FLOOR(a + RAND() * (b - a + 1))
```

Par exemple, cela générera un nombre aléatoire entre 7 et 12

```
SELECT FLOOR(7 + (RAND() * 6));
```

Un moyen simple de retourner aléatoirement les lignes d'une table:

```
SELECT * FROM tbl ORDER BY RAND();
```

Ce sont des nombres **pseudo - aléatoires** .

Le générateur de nombres pseudo-aléatoires de MySQL n'est pas cryptographiquement sécurisé. Autrement dit, si vous utilisez MySQL pour générer des nombres aléatoires à utiliser comme des secrets, un adversaire déterminé qui sait que vous avez utilisé MySQL pourra deviner vos secrets plus facilement que vous ne le pensez.

## Valeur absolue et signe (ABS, SIGN)

Renvoie la valeur absolue d'un nombre

```
SELECT ABS(2);    -> 2
SELECT ABS(-46); -> 46
```

Le `sign` d'un nombre le compare à 0.

| Signe | Résultat | Exemple                                |
|-------|----------|--|
| -1    | $n < 0$  | <code>SELECT SIGN(42); -&gt; 1</code>  |
| 0     | $n = 0$  | <code>SELECT SIGN(0); -&gt; 0</code>   |
| 1     | $n > 0$  | <code>SELECT SIGN(-3); -&gt; -1</code> |

```
SELECT SIGN(-423421); -> -1
```

Lire Arithmétique en ligne: <https://riptutorial.com/fr/mysql/topic/4516/arithmetique>



---

# Chapitre 4: Changer le mot de passe

## Exemples

### Changer le mot de passe root MySQL sous Linux

Pour changer le mot de passe de l'utilisateur root de MySQL:

#### Étape 1: Arrêtez le serveur MySQL.

- dans Ubuntu ou Debian:  
`sudo /etc/init.d/mysql stop`
- dans CentOS, Fedora ou Red Hat Enterprise Linux:  
`sudo /etc/init.d/mysqld stop`

#### Étape 2: Démarrez le serveur MySQL sans le système de privilège.

```
sudo mysqld_safe --skip-grant-tables &
```

ou, si `mysqld_safe` n'est pas disponible,

```
sudo mysqld --skip-grant-tables &
```

#### Étape 3: Connectez - vous au serveur MySQL.

```
mysql -u root
```

#### Étape 4: Définissez un nouveau mot de passe pour l'utilisateur root.

5.7

```
FLUSH PRIVILEGES;  
ALTER USER 'root'@'localhost' IDENTIFIED BY 'new_password';  
FLUSH PRIVILEGES;  
exit;
```

5.7

```
FLUSH PRIVILEGES;  
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');  
FLUSH PRIVILEGES;  
exit;
```

Remarque: La syntaxe `ALTER USER` été introduite dans MySQL 5.7.6.

#### Étape 5: Redémarrez le serveur MySQL.

- dans Ubuntu ou Debian:  
`sudo /etc/init.d/mysql stop`

```
sudo /etc/init.d/mysql start
```

- dans CentOS, Fedora ou Red Hat Enterprise Linux:

```
sudo /etc/init.d/mysqld stop
```

```
sudo /etc/init.d/mysqld start
```

## Modifier le mot de passe root MySQL dans Windows

Lorsque nous voulons changer le mot de passe root dans Windows, nous devons suivre les étapes suivantes:

**Étape 1:** Lancez votre invite de commandes en utilisant l'une des méthodes ci-dessous:

Perss `Ctrl+R` ou `Goto Start Menu > Run`, puis tapez `cmd` et appuyez sur `Entrée`

**Étape 2:** Modifiez votre répertoire en y installant `MYSQL`.

```
C:\> cd C:\mysql\bin
```

**Étape 3:** Maintenant, nous devons démarrer l'invite de commande `mysql`

```
C:\mysql\bin> mysql -u root mysql
```

**Étape 4:** Interroger le feu pour changer `root` mot de passe `root`

```
mysql> SET PASSWORD FOR root@localhost=PASSWORD('my_new_password');
```

## Processus

1. Arrêtez le processus serveur / démon MySQL (`mysqld`).
2. Lancez le traitement du serveur MySQL avec l'option `--skip-grant-tables` pour qu'il ne demande pas de mot de passe: `mysqld_safe --skip-grant-tables &`
3. Connectez-vous au serveur MySQL en tant qu'utilisateur `root`: `mysql -u root`
4. Changer le mot de passe:

- (5.7.6 et plus récents): `ALTER USER 'root'@'localhost' IDENTIFIED BY 'new-password';`
- (5.7.5 et versions ultérieures, ou MariaDB): `SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new-password'); flush privileges; quit;`

5. Redémarrez le serveur MySQL.

Remarque: cela ne fonctionnera que si vous êtes physiquement sur le même serveur.

Doc en ligne: <http://dev.mysql.com/doc/refman/5.7/fr/resetting-permissions.html>

Lire [Changer le mot de passe en ligne](https://riptutorial.com/fr/mysql/topic/2761/changer-le-mot-de-passe): <https://riptutorial.com/fr/mysql/topic/2761/changer-le-mot-de-passe>

# Chapitre 5: CHARGER L'INFILE DE DONNÉES

## Syntaxe

1. LOAD DATA [LOW\_PRIORITY | CONCURRENCE] [LOCAL] INFILE 'nom\_fichier'
2. INTO TABLE nom\_de\_table
3. [CHARACTER SET charset]
4. [{CHAMPS | COLUMNS} [TERMINATED BY 'string'] [[FACULTATIF] ENCLOS PAR 'char']]
5. [LINES [STARTING BY 'string'] [TERMINATED BY 'string']]
6. [Numéro IGNORE {LINES | ROWS}]
7. [(col\_name\_or\_user\_var, ...)]
8. [SET col\_name = expr, ...]

## Exemples

utiliser **LOAD DATA INFILE** pour charger une grande quantité de données dans la base de données

Considérez l'exemple suivant en supposant que vous avez un fichier CSV délimité par un fichier ';' à charger dans votre base de données.

```
1;max;male;manager;12-7-1985
2;jack;male;executive;21-8-1990
.
.
.
1000000;marta;female;accountant;15-6-1992
```

Créez la table pour l'insertion.

```
CREATE TABLE `employee` ( `id` INT NOT NULL ,
                          `name` VARCHAR NOT NULL,
                          `sex` VARCHAR NOT NULL ,
                          `designation` VARCHAR NOT NULL ,
                          `dob` VARCHAR NOT NULL );
```

Utilisez la requête suivante pour insérer les valeurs dans cette table.

```
LOAD DATA INFILE 'path of the file/file_name.txt'
INTO TABLE employee
FIELDS TERMINATED BY ';' //specify the delimiter separating the values
LINES TERMINATED BY '\r\n'
(id,name,sex,designation,dob)
```

Prenons le cas où le format de date est non standard.

```
1;max;male;manager;17-Jan-1985
```

```
2;jack;male;executive;01-Feb-1992
.
.
.
1000000;marta;female;accountant;25-Apr-1993
```

Dans ce cas, vous pouvez changer le format de la colonne `dob` avant de l'insérer comme ceci.

```
LOAD DATA INFILE 'path of the file/file_name.txt'
INTO TABLE employee
FIELDS TERMINATED BY ';' //specify the delimiter separating the values
LINES TERMINATED BY '\r\n'
(id,name,sex,designation,@dob)
SET date = STR_TO_DATE(@date, '%d-%b-%Y');
```

Cet exemple de `LOAD DATA INFILE` ne spécifie pas toutes les fonctionnalités disponibles.

Vous pouvez voir plus de références sur `LOAD DATA INFILE` [ici](#).

## Importer un fichier CSV dans une table MySQL

La commande suivante permet d'importer des fichiers CSV dans une table MySQL avec les mêmes colonnes tout en respectant les règles de citation et d'échappement CSV.

```
load data infile '/tmp/file.csv'
into table my_table
fields terminated by ','
optionally enclosed by '"'
escaped by '\\'
lines terminated by '\n'
ignore 1 lines; -- skip the header row
```

## Charger des données avec des doublons

Si vous utilisez la `LOAD DATA INFILE` pour remplir une table avec des données existantes, vous constaterez souvent que l'importation échoue en raison de doublons. Il existe plusieurs manières possibles de surmonter ce problème.

---

# CHARGER LES DONNÉES LOCALES

Si cette option a été activée sur votre serveur, elle peut être utilisée pour charger un fichier existant sur l'ordinateur client plutôt que sur le serveur. Un effet secondaire est que les lignes en double pour les valeurs uniques sont ignorées.

```
LOAD DATA LOCAL INFILE 'path of the file/file_name.txt'
INTO TABLE employee
```

---

# LOAD DATA INFILE 'fname' REMPLACER

Lorsque le mot-clé `replace` est utilisé, les clés uniques ou primaires dupliquées entraîneront le remplacement de la ligne existante par de nouvelles.

```
LOAD DATA INFILE 'path of the file/file_name.txt'  
REPLACE INTO TABLE employee
```

---

## LOAD DATA INFILE 'fname' IGNORE

À l'inverse de `REPLACE`, les lignes existantes seront conservées et les nouvelles ignorées. Ce comportement est similaire à `LOCAL` décrit ci-dessus. Toutefois, le fichier n'a pas besoin d'exister sur l'ordinateur client.

```
LOAD DATA INFILE 'path of the file/file_name.txt'  
IGNORE INTO TABLE employee
```

---

## Charger via la table intermédiaire

Parfois, ignorer ou remplacer tous les doublons peut ne pas être l'option idéale. Vous devrez peut-être prendre des décisions en fonction du contenu des autres colonnes. Dans ce cas, la meilleure option est de charger dans une table intermédiaire et de transférer à partir de là.

```
INSERT INTO employee SELECT * FROM intermediary WHERE ...
```

### import / export

#### importer

```
SELECT a,b,c INTO OUTFILE 'result.txt' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'  
LINES TERMINATED BY '\n' FROM table;
```

#### Exportation

```
LOAD DATA INFILE 'result.txt' INTO TABLE table;
```

Lire **CHARGER L'INFILE DE DONNÉES** en ligne:

<https://riptutorial.com/fr/mysql/topic/2356/charger-l-infile-de-donnees>

# Chapitre 6: Client MySQL

## Syntaxe

- mysql [OPTIONS] [nom\_base]

## Paramètres

| Paramètre                        | La description   |
|----------------------------------|--|
| -D --database=name               | nom de la base de données  |
| --delimiter=str                  | définir le délimiteur d'instruction. Le par défaut est ';'   |
| -e --execute='command'           | exécuter la commande   |
| -h --host=name                   | nom d'hôte pour se connecter à   |
| -p --password=name               | mot de passe <i>Remarque: il n'y a pas d'espace entre -p et le mot de passe</i>  |
| -p (sans mot de passe)           | le mot de passe vous sera demandé  |
| -P --port=#                      | numéro de port   |
| -s --silent                      | mode silencieux, produire moins de sortie. Utilisez \t comme séparateur de colonne   |
| -ss                              | like -s , mais omet les noms de colonnes   |
| -S --socket=path                 | spécifier le socket (Unix) ou le canal nommé (Windows) à utiliser lors de la connexion à une instance locale   |
| --skip-column-names              | omettre les noms de colonnes   |
| -u --user=name                   | Nom d'utilisateur  |
| -U --safe-updates --i-am-a-dummy | connectez-vous avec la variable <code>sql_safe_updates=ON</code> . Cela permettra uniquement <code>DELETE</code> et <code>UPDATE</code> qui utilisent explicitement les clés |
| -V --version                     | imprimer la version et quitter   |

## Exemples

## Connexion de base

Pour accéder à MySQL depuis la ligne de commande:

```
mysql --user=username --password=pwd --host=hostname test_db
```

Cela peut être raccourci à:

```
mysql -u username -p password -h hostname test_db
```

En omettant la valeur du `password`, MySQL demandera le mot de passe requis comme première entrée. Si vous spécifiez un `password` le client vous enverra un avertissement «non sécurisé»:

```
mysql -u=username -p -h=hostname test_db
```

Pour les connexions locales `--socket` peut être utilisé pour pointer vers le fichier socket:

```
mysql --user=username --password=pwd --host=localhost --socket=/path/to/mysqld.sock test_db
```

Si vous omettez le paramètre `socket`, le client tentera de se connecter à un serveur sur l'ordinateur local. Le serveur doit être en cours d'exécution pour s'y connecter.

## Exécuter des commandes

Cet ensemble d'exemples montre comment exécuter des commandes stockées dans des chaînes ou des fichiers de script, sans avoir besoin de l'invite interactive. Ceci est particulièrement utile lorsqu'un script shell doit interagir avec une base de données.

## Exécuter la commande à partir d'une chaîne

```
$ mysql -uroot -proot test -e'select * from people'
```

```
+----+-----+-----+
| id | name  | gender |
+----+-----+-----+
|  1 | Kathy | f      |
|  2 | John  | m      |
+----+-----+-----+
```

Pour formater la sortie en tant que grille séparée par des tabulations, utilisez le paramètre `--silent`:

```
$ mysql -uroot -proot test -s -e'select * from people'
```

```
id      name      gender
1       Kathy     f
2       John      m
```

Pour omettre les en-têtes:

```
$ mysql -uroot -proot test -ss -e'select * from people'
```

```
1      Kathy  f
2      John   m
```

---

## Exécuter à partir du fichier script:

```
$ mysql -uroot -proot test < my_script.sql
```

```
$ mysql -uroot -proot test -e'source my_script.sql'
```

---

## Ecrire la sortie sur un fichier

```
$ mysql -uroot -proot test < my_script.sql > out.txt
```

```
$ mysql -uroot -proot test -s -e'select * from people' > out.txt
```

Lire Client MySQL en ligne: <https://riptutorial.com/fr/mysql/topic/5619/client-mysql>



---

# Chapitre 7: Codes d'erreur

## Exemples

### Code d'erreur 1064: erreur de syntaxe

```
select LastName, FirstName,  
from Person
```

Renvoie un message:

Code d'erreur: 1064. Vous avez une erreur dans votre syntaxe SQL; Consultez le manuel correspondant à votre version du serveur MySQL pour connaître la syntaxe appropriée à utiliser près de 'from Person' à la ligne 2.

Obtenir un message d'erreur "1064" de MySQL signifie que la requête ne peut pas être analysée sans erreurs de syntaxe. En d'autres termes, il ne peut pas donner de sens à la requête.

La citation dans le message d'erreur commence par le premier caractère de la requête que MySQL ne peut pas comprendre. Dans cet exemple, MySQL n'a pas de sens, en contexte, `from Person`. Dans ce cas, il y a une virgule supplémentaire immédiatement avant `from Person`. La virgule dit à MySQL d'attendre une autre description de colonne dans la clause `SELECT`

Une erreur de syntaxe dit toujours `... near '...'`. La chose au début des guillemets est très proche de l'erreur. Pour localiser une erreur, examinez le premier jeton entre guillemets et le dernier jeton avant les guillemets.

Parfois, vous allez `... near ''` c'est-à-dire rien dans les citations. Cela signifie que le premier caractère que MySQL ne comprend pas est à la fin ou au début de l'instruction. Cela suggère que la requête contient des citations non équilibrées ( ' ou " ) ou des parenthèses non équilibrées ou que vous n'avez pas terminé l'instruction correctement avant.

Dans le cas d'une routine stockée, vous avez peut-être oublié d'utiliser `DELIMITER`.

Ainsi, lorsque vous obtenez l'erreur 1064, examinez le texte de la requête et recherchez le point mentionné dans le message d'erreur. Inspectez visuellement le texte de la requête juste autour de ce point.

Si vous demandez à quelqu'un de vous aider à résoudre l'erreur 1064, il est préférable de fournir à la fois le texte de la requête entière et le texte du message d'erreur.

### Code d'erreur 1175: Mise à jour sécurisée

Cette erreur apparaît lors de la tentative de mise à jour ou de suppression des enregistrements sans inclure la clause `WHERE` qui utilise la colonne `KEY`.

Pour exécuter la suppression ou la mise à jour de toute façon - tapez:

```
SET SQL_SAFE_UPDATES = 0;
```

Pour réactiver le mode sans échec - tapez:

```
SET SQL_SAFE_UPDATES = 1;
```

## Code d'erreur 1215: Impossible d'ajouter une contrainte de clé étrangère

Cette erreur se produit lorsque les tables ne sont pas structurées de manière adéquate pour gérer la vérification rapide des exigences de clé étrangère (FK) que le développeur impose.

```
CREATE TABLE `gtType` (  
  `type` char(2) NOT NULL,  
  `description` varchar(1000) NOT NULL,  
  PRIMARY KEY (`type`)  
) ENGINE=InnoDB;  
  
CREATE TABLE `getTogethers` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `type` char(2) NOT NULL,  
  `eventDT` datetime NOT NULL,  
  `location` varchar(1000) NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `fk_gt2type` (`type`), -- see Note1 below  
  CONSTRAINT `gettogethers_ibfk_1` FOREIGN KEY (`type`) REFERENCES `gtType` (`type`)  
) ENGINE=InnoDB;
```

Note 1: une KEY comme celle-ci sera créée automatiquement si nécessaire en raison de la définition FK dans la ligne qui la suit. Le développeur peut l'ignorer et le KEY (aka index) sera ajouté si nécessaire. Un exemple d'évitement par le développeur est montré ci-dessous dans une `someOther`.

Jusqu'ici tout va bien, jusqu'à l'appel ci-dessous.

```
CREATE TABLE `someOther` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `someDT` datetime NOT NULL,  
  PRIMARY KEY (`id`),  
  CONSTRAINT `someOther_dt` FOREIGN KEY (`someDT`) REFERENCES `getTogethers` (`eventDT`)  
) ENGINE=InnoDB;
```

### Code d'erreur: 1215. Impossible d'ajouter une contrainte de clé étrangère

Dans ce cas, il échoue en raison de l'absence d'index dans la table *référéncée* `getTogethers` pour gérer la recherche rapide d'un `eventDT`. À résoudre dans la prochaine déclaration.

```
CREATE INDEX `gt_eventdt` ON getTogethers (`eventDT`);
```

La table `getTogethers` a été modifiée et maintenant la création de `someOther` réussira.

À partir de la page du manuel MySQL [Utilisation des contraintes FOREIGN KEY](#) :

MySQL nécessite des index sur les clés étrangères et les clés référencées afin que les vérifications de clés étrangères puissent être rapides et ne nécessitent pas d'analyse de table. Dans la table de référence, il doit y avoir un index où les colonnes de la clé étrangère sont répertoriées comme les premières colonnes du même ordre. Un tel index est créé automatiquement sur la table de référence s'il n'existe pas.

Les colonnes correspondantes dans la clé étrangère et la clé référencée doivent avoir des types de données similaires. La taille et le signe des types entiers doivent être identiques. La longueur des types de chaîne ne doit pas nécessairement être la même. Pour les colonnes de chaîne non binaires (caractères), le jeu de caractères et le classement doivent être identiques.

InnoDB permet à une clé étrangère de référencer une colonne d'index ou un groupe de colonnes. Toutefois, dans la table référencée, il doit y avoir un index où les colonnes référencées sont répertoriées comme les premières colonnes du même ordre.

Notez le dernier point ci-dessus concernant les premières colonnes (les plus à gauche) et l'absence de condition de clé primaire (bien que fortement recommandée).

Après la création d'une table de *référence* (enfant), toutes les clés créées automatiquement pour vous sont visibles avec une commande telle que:

```
SHOW CREATE TABLE someOther;
```

Parmi les autres cas courants d'expérience de cette erreur, citons les documents ci-dessus, mais ils doivent être mis en évidence:

- Différences apparemment insignifiantes de l' `INT` qui sont signées, pointant vers `INT UNSIGNED`.
- Les développeurs ont du mal à comprendre les KEYS multi-colonnes (composites) et les premières exigences de commande (les plus à gauche).

## 1045 Accès refusé

Voir les discussions dans "GRANT" et "Récupérer le mot de passe root".

## 1236 "position impossible" dans la réplication

*Habituellement*, cela signifie que le Master est tombé en panne et que `sync_binlog` était OFF. La solution est de `CHANGE MASTER to POS=0` du fichier binlog suivant (voir le fichier maître) sur l'esclave.

La cause: le maître envoie les éléments de réplication à l'esclave avant de vider son binlog (quand `sync_binlog=OFF`). Si le Master se bloque avant le flush, l'esclave a déjà logiquement dépassé la fin du fichier sur le binlog. Lorsque le Master redémarre, il lance un nouveau binlog. La meilleure solution est donc de changer au début de ce binlog.

Une solution à plus long terme est `sync_binlog=ON`, si vous pouvez vous permettre les E / S

supplémentaires `sync_binlog=ON` .

(Si vous utilisez GTID, ...?)

## 2002, 2003 Impossible de se connecter

Recherchez un problème bloquant le port 3306.

Quelques diagnostics et / ou solutions possibles

- Le serveur est-il en cours d'exécution?
- "service firewalld stop" et "systemctl disable firewalld"
- telnet master 3306
- Vérifiez l' `bind-address`
- vérifiez le `skip-name-resolve`
- vérifiez la prise.

## 1067, 1292, 1366, 1411 - Mauvaise valeur pour le nombre, la date, la valeur par défaut, etc.

**1067** Ceci est probablement lié aux valeurs par défaut `TIMESTAMP` , qui ont changé au fil du temps. Voir `TIMESTAMP defaults` dans la page Dates et heures. (qui n'existe pas encore)

**1292/1366 DOUBLE / Integer Recherche les lettres** ou autres erreurs de syntaxe. Vérifiez que les colonnes sont alignées. vous pensez peut-être que vous mettez dans un `VARCHAR` mais il est aligné avec une colonne numérique.

**1292 DATETIME** Vérifiez trop loin dans le passé ou le futur. Vérifiez entre 2h et 3h du matin lorsque l'heure avancée a changé. Vérifiez la syntaxe incorrecte, telle que `+00` éléments de fuseau horaire.

**1292 VARIABLE** Vérifiez les valeurs autorisées pour la `VARIABLE` vous essayez de `SET` .

**1292 LOAD DATA** Regardez la ligne qui est "mauvaise". Vérifiez les symboles d'échappement, etc. Regardez les types de données.

**1411 STR\_TO\_DATE Date** incorrectement formatée?

## 126, 127, 134, 144, 145

Lorsque vous essayez d'accéder aux enregistrements de la base de données MySQL, vous pouvez obtenir ces messages d'erreur. Ces messages d'erreur sont dus à une corruption de la base de données MySQL. Voici les types

```
MySQL error code 126 = Index file is crashed
MySQL error code 127 = Record-file is crashed
MySQL error code 134 = Record was already deleted (or record file crashed)
MySQL error code 144 = Table is crashed and last repair failed
MySQL error code 145 = Table was marked as crashed and should be repaired
```

Le bogue MySQL, les attaques de virus, les pannes de serveur, les arrêts incorrects, les tables endommagées sont les raisons de cette corruption. Quand il est corrompu, il devient inaccessible et vous ne pouvez plus y accéder. Afin d'obtenir l'accessibilité, le meilleur moyen de récupérer des données à partir d'une sauvegarde mise à jour. Cependant, si vous ne disposez d'aucune sauvegarde valide ou mise à jour, vous pouvez opter pour MySQL Repair.

Si le type de moteur de table est `MyISAM`, appliquez `CHECK TABLE`, puis `REPAIR TABLE` à celui-ci.

Alors réfléchissez sérieusement à la conversion à InnoDB, donc cette erreur ne se reproduira plus.

## Syntaxe

```
CHECK TABLE <table name> ////To check the extent of database corruption
REPAIR TABLE <table name> ////To repair table
```

## 139

L'erreur 139 peut signifier que le nombre et la taille des champs dans la définition de la table dépasse une certaine limite. Solutions de contournement:

- Repenser le schéma
- Normaliser certains champs
- Partitionner verticalement la table

## 1366

Cela signifie généralement que la gestion du jeu de caractères n'était pas cohérente entre le client et le serveur. Voir ... pour plus d'assistance.

## 126, 1054, 1146, 1062, 24

(en prenant une pause) Avec l'inclusion de ces 4 numéros d'erreur, je pense que cette page aura couvert environ 50% des erreurs typiques que les utilisateurs obtiennent.

(Oui, cet exemple doit être révisé.)

## 24 Impossible d'ouvrir le fichier (Trop de fichiers ouverts)

`open_files_limit` provient d'un paramètre du système d'exploitation. `table_open_cache` doit être inférieur à cela.

Celles-ci peuvent provoquer cette erreur:

- Échec de `DEALLOCATE PREPARE` dans une procédure stockée.
- Table (s) `PARTITIONed` avec un grand nombre de partitions et `innodb_file_per_table = ON`. Recommande de ne pas avoir plus de 50 partitions dans une table donnée (pour diverses raisons). (Lorsque "Native Partitions" devient disponible, ce conseil peut changer.)

La solution évidente consiste à définir augmenter la limite du système d'exploitation: Pour permettre à plus de fichiers, modifier `ulimit` ou `/etc/security/limits.conf` ou `sysctl.conf` (`kern.maxfiles` & `kern.maxfilesperproc`) ou autre chose (OS à charge). Augmentez ensuite `open_files_limit` et `table_open_cache`.

A partir de 5.6.8, `open_files_limit` est automatiquement dimensionné en fonction de `max_connections`, mais il est correct de le modifier par défaut.

## 1062 - Entrée en double

Cette erreur se produit principalement pour les deux raisons suivantes

### 1. Valeur en double - Error Code: 1062. Duplicate entry '12' for key 'PRIMARY'

La colonne de clé primaire est unique et n'acceptera pas l'entrée en double. Donc, lorsque vous essayez d'insérer une nouvelle ligne qui est déjà présente dans votre table, cela produira cette erreur.

Pour résoudre ce problème, définissez la colonne de clé primaire comme `AUTO_INCREMENT`. Et lorsque vous essayez d'insérer une nouvelle ligne, ignorez la colonne de clé primaire ou insérez la valeur `NULL` dans la clé primaire.

```
CREATE TABLE userDetails(  
  userId INT(10) NOT NULL AUTO_INCREMENT,  
  firstName VARCHAR(50),  
  lastName VARCHAR(50),  
  isActive INT(1) DEFAULT 0,  
  PRIMARY KEY (userId) );  
  
--->and now while inserting  
INSERT INTO userDetails VALUES (NULL, 'John', 'Doe', 1);
```

### 2. Champ de données unique - Error Code: 1062. Duplicate entry 'A' for key 'code'

Vous pouvez assigner une colonne unique et essayer d'insérer une nouvelle ligne avec une valeur déjà existante pour cette colonne produira cette erreur.

Pour surmonter cette erreur, utilisez `INSERT IGNORE` au lieu de `INSERT` normal. Si la nouvelle ligne que vous essayez d'insérer ne duplique pas un enregistrement existant, MySQL l'insère comme d'habitude. Si l'enregistrement est un doublon, le mot clé `IGNORE` le `IGNORE` sans générer d'erreur.

```
INSERT IGNORE INTO userDetails VALUES (NULL, 'John', 'Doe', 1);
```

Lire Codes d'erreur en ligne: <https://riptutorial.com/fr/mysql/topic/895/codes-d-erreur>

# Chapitre 8: COMMANDÉ PAR

## Exemples

### Les contextes

Les clauses d'un `SELECT` ont un ordre spécifique:

```
SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ...
    ORDER BY ... -- goes here
    LIMIT ... OFFSET ...;

( SELECT ... ) UNION ( SELECT ... ) ORDER BY ... -- for ordering the result of the UNION.

SELECT ... GROUP_CONCAT(DISTINCT x ORDER BY ... SEPARATOR ...) ...

ALTER TABLE ... ORDER BY ... -- probably useful only for MyISAM; not for InnoDB
```

### De base

#### COMMANDER PAR x

x peut être n'importe quel type de données.

- `NULLs` précède les non-`NULL`.
- La valeur par défaut est `ASC` (du plus bas au plus élevé)
- Les chaînes ( `VARCHAR` , etc.) sont classées selon la `COLLATION` de la déclaration
- `ENUMs` sont classés par ordre de déclaration de ses chaînes.

#### Ascendant descendant

```
ORDER BY x ASC -- same as default
ORDER BY x DESC -- highest to lowest
ORDER BY lastname, firstname -- typical name sorting; using two columns
ORDER BY submit_date DESC -- latest first
ORDER BY submit_date DESC, id ASC -- latest first, but fully specifying order.
```

- `ASC = ASCENDING` , `DESC = DESCENDING`
- `NULLs` vient en premier, même pour `DESC` .
- Dans les exemples ci-dessus, `INDEX(x)` , `INDEX(lastname, firstname)` , `INDEX(submit_date)` peuvent considérablement améliorer les performances.

Mais ... Mélanger `ASC` et `DESC` , comme dans le dernier exemple, ne peut pas utiliser un index composite pour en bénéficier. `INDEX(submit_date DESC, id ASC)` n'aidera pas non plus - " `DESC` " est reconnu syntaxiquement dans la déclaration `INDEX` , mais est ignoré.

#### Des astuces

```
ORDER BY FIND_IN_SET(card_type, "MASTER-CARD,VISA,DISCOVER") -- sort 'MASTER-CARD' first.  
ORDER BY x IS NULL, x -- order by `x`, but put `NULLs` last.
```

## Commande personnalisée

```
SELECT * FROM some_table WHERE id IN (118, 17, 113, 23, 72)  
ORDER BY FIELD(id, 118, 17, 113, 23, 72);
```

Renvoie le résultat dans l'ordre spécifié des identifiants.

| id  | ... |
|-----|-----|
| 118 | ... |
| 17  | ... |
| 113 | ... |
| 23  | ... |
| 72  | ... |

Utile si les identifiants sont déjà triés et que vous avez juste besoin de récupérer les lignes.

Lire **COMMANDÉ PAR** en ligne: <https://riptutorial.com/fr/mysql/topic/5469/commande-par>



---

# Chapitre 9: Commentaire Mysql

## Remarques

Le -- style de commentaire, ce qui nécessite un espace de fuite, [diffère dans le comportement de la norme SQL](#) , qui ne nécessite pas l'espace.

## Exemples

### Ajouter des commentaires

Il existe trois types de commentaire:

```
# This comment continues to the end of line

-- This comment continues to the end of line

/* This is an in-line comment */

/*
This is a
multiple-line comment
*/
```

Exemple:

```
SELECT * FROM t1; -- this is comment

CREATE TABLE stack(
  /*id_user int,
  username varchar(30),
  password varchar(30)
  */
  id int
);
```

La méthode -- exige qu'un espace suive le -- avant que le commentaire ne commence, sinon il sera interprété comme une commande et provoquera généralement une erreur.

```
#This comment works
/*This comment works.*/
--This comment does not.
```

### Définitions des tables de commentaires

```
CREATE TABLE menagerie.bird (
  bird_id INT NOT NULL AUTO_INCREMENT,
  species VARCHAR(300) DEFAULT NULL COMMENT 'You can include genus, but never subspecies.',
  INDEX idx_species (species) COMMENT 'We must search on species often.',
```

```
PRIMARY KEY (bird_id)
) ENGINE=InnoDB COMMENT 'This table was inaugurated on February 10th.';
```

En utilisant un = après `COMMENT` est facultative. ( [Documents officiels](#) )

Ces commentaires, contrairement aux autres, sont enregistrés avec le schéma et peuvent être récupérés via `SHOW CREATE TABLE` ou `from information_schema` .

Lire [Commentaire Mysql en ligne](#): <https://riptutorial.com/fr/mysql/topic/2337/commentaire-mysql>

---

# Chapitre 10: Configuration de la connexion SSL

## Exemples

### Configuration pour les systèmes basés sur Debian

(Cela suppose que MySQL a été installé et que `sudo` est utilisé.)

---

## Génération d'une clé CA et d'une clé SSL

Assurez-vous que OpenSSL et les bibliothèques sont installés:

```
apt-get -y install openssl
apt-get -y install libssl-dev
```

Faites ensuite et entrez un répertoire pour les fichiers SSL:

```
mkdir /home/ubuntu/mysqlcerts
cd /home/ubuntu/mysqlcerts
```

Pour générer des clés, créez une autorité de certification (CA) pour signer les clés (auto-signées):

```
openssl genrsa 2048 > ca-key.pem
openssl req -new -x509 -nodes -days 3600 -key ca-key.pem -out ca.pem
```

Les valeurs entrées à chaque invite n'affecteront pas la configuration. Ensuite, créez une clé pour le serveur et signez avec l'autorité de certification d'avant:

```
openssl req -newkey rsa:2048 -days 3600 -nodes -keyout server-key.pem -out server-req.pem
openssl rsa -in server-key.pem -out server-key.pem

openssl x509 -req -in server-req.pem -days 3600 -CA ca.pem -CAkey ca-key.pem -set_serial 01 -
out server-cert.pem
```

Créez ensuite une clé pour un client:

```
openssl req -newkey rsa:2048 -days 3600 -nodes -keyout client-key.pem -out client-req.pem
openssl rsa -in client-key.pem -out client-key.pem
openssl x509 -req -in client-req.pem -days 3600 -CA ca.pem -CAkey ca-key.pem -set_serial 01 -
out client-cert.pem
```

Pour vous assurer que tout a été configuré correctement, vérifiez les clés:

```
openssl verify -CAfile ca.pem server-cert.pem client-cert.pem
```

---

# Ajouter les clés à MySQL

Ouvrez le [fichier de configuration MySQL](#) . Par exemple:

```
vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

Sous la section `[mysqld]` , ajoutez les options suivantes:

```
ssl-ca = /home/ubuntu/mysqlcerts/ca.pem
ssl-cert = /home/ubuntu/mysqlcerts/server-cert.pem
ssl-key = /home/ubuntu/mysqlcerts/server-key.pem
```

Redémarrez MySQL. Par exemple:

```
service mysql restart
```

---

# Tester la connexion SSL

Connectez-vous de la même manière en transmettant les options supplémentaires `ssl-ca` , `ssl-cert` et `ssl-key` à l'aide de la clé client générée. Par exemple, en supposant que `cd /home/ubuntu/mysqlcerts :`

```
mysql --ssl-ca=ca.pem --ssl-cert=client-cert.pem --ssl-key=client-key.pem -h 127.0.0.1 -u
superman -p
```

Une fois connecté, vérifiez que la connexion est bien sécurisée:

```
superman@127.0.0.1 [None]> SHOW VARIABLES LIKE '%ssl%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| have_openssl  | YES   |
| have_ssl      | YES   |
| ssl_ca        | /home/ubuntu/mysqlcerts/ca.pem |
| ssl_capath    |       |
| ssl_cert      | /home/ubuntu/mysqlcerts/server-cert.pem |
| ssl_cipher    |       |
| ssl_crl       |       |
| ssl_crlpath   |       |
| ssl_key       | /home/ubuntu/mysqlcerts/server-key.pem |
+-----+-----+
```

Vous pouvez également vérifier:

```
superman@127.0.0.1 [None]> STATUS;
...
SSL:                Cipher in use is DHE-RSA-AES256-SHA
...
```

# Application du protocole SSL

Ceci est via `GRANT` , en utilisant `REQUIRE SSL` :

```
GRANT ALL PRIVILEGES ON *.* TO 'superman'@'127.0.0.1' IDENTIFIED BY 'pass' REQUIRE SSL;  
FLUSH PRIVILEGES;
```

Maintenant, `superman` *doit se connecter* via SSL.

Si vous ne souhaitez pas gérer les clés client, utilisez la clé client plus tôt et utilisez-la automatiquement pour tous les clients. Ouvrez le [fichier de configuration MySQL](#) , par exemple:

```
vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

Sous la section `[client]` , ajoutez les options suivantes:

```
ssl-ca = /home/ubuntu/mysqlcerts/ca.pem  
ssl-cert = /home/ubuntu/mysqlcerts/client-cert.pem  
ssl-key = /home/ubuntu/mysqlcerts/client-key.pem
```

Maintenant, `superman` doit seulement saisir ce qui suit pour se connecter via SSL:

```
mysql -h 127.0.0.1 -u superman -p
```

La connexion à partir d'un autre programme, par exemple en Python, ne nécessite généralement qu'un paramètre supplémentaire pour la fonction de connexion. Un exemple Python:

```
import MySQLdb  
ssl = {'cert': '/home/ubuntu/mysqlcerts/client-cert.pem', 'key':  
'/home/ubuntu/mysqlcerts/client-key.pem'}  
conn = MySQLdb.connect(host='127.0.0.1', user='superman', passwd='imsoawesome', ssl=ssl)
```

## Références et lectures complémentaires:

- <https://www.percona.com/blog/2013/06/22/setting-up-mysql-ssl-and-secure-connections/>
- <https://lowendbox.com/blog/getting-started-with-mysql-over-ssl/>
- <http://xmodulo.com/enable-ssl-mysql-server-client.html>
- <https://ubuntuforums.org/showthread.php?t=1121458>

## Configuration pour CentOS7 / RHEL7

Cet exemple suppose deux serveurs:

1. dbserver (où réside notre base de données)
2. appclient (où nos applications vivent)

*FWIW, les deux serveurs sont forcés par SELinux.*

# Tout d'abord, connectez-vous à dbserver

Créez un répertoire temporaire pour créer les certificats.

```
mkdir /root/certs/mysql/ && cd /root/certs/mysql/
```

## Créer les certificats du serveur

```
openssl genrsa 2048 > ca-key.pem
openssl req -sha1 -new -x509 -nodes -days 3650 -key ca-key.pem > ca-cert.pem
openssl req -sha1 -newkey rsa:2048 -days 730 -nodes -keyout server-key.pem > server-req.pem
openssl rsa -in server-key.pem -out server-key.pem
openssl x509 -sha1 -req -in server-req.pem -days 730 -CA ca-cert.pem -CAkey ca-key.pem -
set_serial 01 > server-cert.pem
```

Déplacer les certificats de serveur vers / etc / pki / tls / certs / mysql /

Le chemin de répertoire suppose CentOS ou RHEL (ajustez au besoin pour les autres distributions):

```
mkdir /etc/pki/tls/certs/mysql/
```

Veillez à définir des autorisations sur le dossier et les fichiers. mysql a besoin de la pleine propriété et de l'accès.

```
chown -R mysql:mysql /etc/pki/tls/certs/mysql
```

## Configurez maintenant MySQL / MariaDB

```
# vi /etc/my.cnf
# i
[mysqld]
bind-address=*
ssl-ca=/etc/pki/tls/certs/ca-cert.pem
ssl-cert=/etc/pki/tls/certs/server-cert.pem
ssl-key=/etc/pki/tls/certs/server-key.pem
# :wq
```

alors

```
systemctl restart mariadb
```

N'oubliez pas d'ouvrir votre pare-feu pour autoriser les connexions depuis applient (en utilisant IP 1.2.3.4)

```
firewall-cmd --zone=drop --permanent --add-rich-rule 'rule family="ipv4" source
address="1.2.3.4" service name="mysql" accept'
# I force everything to the drop zone. Season the above command to taste.
```

Maintenant, redémarrez firewalld

```
service firewalld restart
```

Ensuite, connectez-vous au serveur mysql de observer:

```
mysql -uroot -p
```

Émettez ce qui suit pour créer un utilisateur pour le client. notez SSL OBLIGATOIRE dans l'instruction GRANT.

```
GRANT ALL PRIVILEGES ON *.* TO 'iamsecure'@'appclient' IDENTIFIED BY 'dingdingding' REQUIRE  
SSL;  
FLUSH PRIVILEGES;  
# quit mysql
```

Vous devriez toujours être dans / root / certs / mysql dès la première étape. Si non, cd y revenir pour une des commandes ci-dessous.

Créer les certificats clients

```
openssl req -sha1 -newkey rsa:2048 -days 730 -nodes -keyout client-key.pem > client-req.pem  
openssl rsa -in client-key.pem -out client-key.pem  
openssl x509 -sha1 -req -in client-req.pem -days 730 -CA ca-cert.pem -CAkey ca-key.pem -  
set_serial 01 > client-cert.pem
```

**Remarque** : j'ai utilisé le même nom commun pour les certificats serveur et client. YMMV.

*Assurez-vous que vous êtes toujours / root / certs / mysql / pour cette prochaine commande*

Combinez un certificat de CA serveur et client dans un seul fichier:

```
cat server-cert.pem client-cert.pem > ca.pem
```

Assurez-vous de voir deux certificats:

```
cat ca.pem
```

---

## FIN DU SERVEUR CÔTÉ TRAVAIL POUR MAINTENANT.

Ouvrez un autre terminal et

```
ssh appclient
```

Comme auparavant, créer un domicile permanent pour les certificats clients

```
mkdir /etc/pki/tls/certs/mysql/
```

Maintenant, placez les certificats clients (créés sur dbserver) sur appclient. Vous pouvez soit les parcourir, soit copier et coller les fichiers un par un.

```
scp dbserver
# copy files from dbserver to appclient
# exit scp
```

Encore une fois, veillez à définir des autorisations sur le dossier et les fichiers. mysql a besoin de la pleine propriété et de l'accès.

```
chown -R mysql:mysql /etc/pki/tls/certs/mysql
```

Vous devriez avoir trois fichiers, chacun appartenant à l'utilisateur mysql:

```
/etc/pki/tls/certs/mysql/ca.pem
/etc/pki/tls/certs/mysql/client-cert.pem
/etc/pki/tls/certs/mysql/client-key.pem
```

Maintenant, éditez la configuration MariaDB / MySQL de l'appclient dans la section `[client]`.

```
vi /etc/my.cnf
# i
[client]
ssl-ca=/etc/pki/tls/certs/mysql/ca.pem
ssl-cert=/etc/pki/tls/certs/mysql/client-cert.pem
ssl-key=/etc/pki/tls/certs/mysql/client-key.pem
# :wq
```

Redémarrez le service mariadb de l'appclient:

```
systemctl restart mariadb
```

---

## toujours sur le client ici

Cela devrait retourner: ssl TRUE

```
mysql --ssl --help
```

Connectez-vous maintenant à l'instance mysql de l'appclient

```
mysql -uroot -p
```

Devrait voir OUI aux deux variables ci-dessous

```
show variables LIKE '%ssl';
have_openssl      YES
```



```
have_ssl YES
```

Au départ, j'ai vu

```
have_openssl NO
```

Un rapide coup d'œil sur mariadb.log a révélé:

```
Erreur SSL: impossible d'obtenir le certificat à partir de /etc/pki/tls/certs/mysql/client-cert.pem
```

Le problème était que la racine possédait client-cert.pem et le dossier contenant. La solution consistait à définir la propriété de / etc / pki / tls / certs / mysql / sur mysql.

```
chown -R mysql:mysql /etc/pki/tls/certs/mysql
```

Redémarrez mariadb si nécessaire à partir de l'étape ci-dessus

---

## MAINTENANT, NOUS SOMMES PRÊTS À TESTER LA CONNEXION SÉCURISÉE

---

### Nous sommes toujours sur appclient ici

Essayez de vous connecter à l'instance mysql de dbserver en utilisant le compte créé ci-dessus.

```
mysql -h dbserver -u iamsecure -p
# enter password dingdingding (hopefully you changed that to something else)
```

Avec un peu de chance, vous devriez être connecté sans erreur.

Pour confirmer que vous êtes connecté avec SSL activé, lancez la commande suivante à partir de l'invite MariaDB / MySQL:

```
\s
```

*C'est un backslash, alias statut*

Cela montrera le statut de votre connexion, qui devrait ressembler à ceci:

```
Connection id:          4
Current database:
Current user:           iamsecure@appclient
SSL:                   Cipher in use is DHE-RSA-AES256-GCM-SHA384
Current pager:         stdout
Using outfile:         ''
Using delimiter:       ;
```

```
Server:          MariaDB
Server version:  5.X.X-MariaDB MariaDB Server
Protocol version: 10
Connection:      dbserver via TCP/IP
Server charset:  latin1
Db charset:      latin1
Client charset:  utf8
Conn. charset:   utf8
TCP port:        3306
Uptime:          42 min 13 sec
```

Si vous obtenez une autorisation refusée pour des erreurs sur votre tentative de connexion, vérifiez votre instruction GRANT ci-dessus pour vous assurer qu'il n'y a pas de caractères ou de marques parasites.

Si vous avez des erreurs SSL, consultez ce guide pour vous assurer que les étapes sont correctes.

Cela a fonctionné sur RHEL7 et fonctionnera probablement aussi sur CentOS7. Impossible de confirmer si ces étapes exactes fonctionneront ailleurs.

J'espère que cela sauve quelqu'un d'autre un peu de temps et d'aggravation.

Lire [Configuration de la connexion SSL en ligne](https://riptutorial.com/fr/mysql/topic/7563/configuration-de-la-connexion-ssl):

<https://riptutorial.com/fr/mysql/topic/7563/configuration-de-la-connexion-ssl>

---

# Chapitre 11: Configuration et réglage

## Remarques

La configuration se fait de trois manières différentes:

- options de ligne de commande
- le fichier de configuration `my.cnf`
- définir des variables depuis le serveur

Les options de la ligne de commande prennent la forme `mysqld --long-parameter-name=value --another-parameter`. Les mêmes paramètres peuvent être placés dans le fichier de configuration `my.cnf`. Certains paramètres sont configurables à l'aide de variables système provenant de MySQL. Consultez la documentation officielle pour une liste complète des paramètres.

Les variables peuvent avoir trait `-` ou souligner `_`. Des espaces peuvent exister autour du `=`. Les grands nombres peuvent être suffixés par `K`, `M`, `G` pour kilo-, mega et giga-. Un réglage par ligne.

Drapeaux: Habituellement, `ON` et `1` sont synonymes, idem pour `OFF` et `0`. Certains drapeaux n'ont rien après eux.

Lorsque vous placez les paramètres dans `my.cnf`, tous les paramètres du *serveur* doivent se trouver dans la section `[mysqld]` donc pas aveuglement des paramètres à la fin du fichier. (Remarque: Pour les outils permettant à plusieurs instances de mysql de partager un fichier `my.cnf`, les noms de section peuvent être différents.)

## Exemples

### Performance InnoDB

Des centaines de paramètres peuvent être placés dans `my.cnf`. Pour l'utilisateur "lite" de MySQL, cela n'aura pas beaucoup d'importance.

Une fois que votre base de données devient non triviale, il est conseillé de définir les paramètres suivants:

```
innodb_buffer_pool_size
```

Cela devrait être réglé à environ 70% de la RAM *disponible* (si vous avez au moins 4 Go de RAM, un pourcentage plus petit si vous avez une petite VM ou une machine ancienne). Le paramètre contrôle la quantité de cache utilisée par InnoDB ENGINE. Par conséquent, il est très important pour la performance d'InnoDB.

### Paramètre permettant l'insertion de données volumineuses

Si vous avez besoin de stocker des images ou des vidéos dans la colonne, nous devons modifier

la valeur en fonction de votre application.

```
max_allowed_packet = 10M
```

M est Mb, G en Gb, K en Kb

## Augmenter la limite de chaîne pour `group_concat`

`group_concat` permet de concaténer des valeurs non nulles dans un `group`. La longueur maximale de la chaîne résultante peut être définie à l'aide de l'option `group_concat_max_len` :

```
SET [GLOBAL | SESSION] group_concat_max_len = val;
```

La définition de la variable `GLOBAL` assurera un changement permanent, tandis que la définition de la variable `SESSION` définira la valeur pour la session en cours.

## Configuration minimale d'InnoDB

C'est une configuration minimale pour les serveurs MySQL utilisant des tables InnoDB. En utilisant InnoDB, le cache de requêtes n'est pas requis. Récupérer de l'espace disque lorsqu'une table ou une base de données est `DROP` ed. Si vous utilisez des disques SSD, le vidage est une opération redondante (les SDD ne sont pas séquentiels).

```
default_storage_engine = InnoDB
query_cache_type = 0
innodb_file_per_table = 1
innodb_flush_neighbors = 0
```

## Concurrence

Assurez-vous de pouvoir créer plus que les 4 threads par défaut en définissant `innodb_thread_concurrency` à l'infini (0); Cela permet à InnoDB de décider en fonction d'une exécution optimale.

```
innodb_thread_concurrency = 0
innodb_read_io_threads = 64
innodb_write_io_threads = 64
```

## Utilisation du disque dur

Définissez la capacité (charge normale) et la capacité\_max (maximum absolu) d'IOPS pour MySQL. La valeur par défaut de 200 est correcte pour les disques durs, mais ces jours-ci, avec des disques SSD capables de milliers d'IOPS, vous souhaitez probablement ajuster ce nombre. Il existe de nombreux tests que vous pouvez exécuter pour déterminer les IOPS. Les valeurs ci-dessus devraient être presque la même *si vous utilisez un serveur MySQL dédié*. Si vous exécutez d'autres services sur le même ordinateur, vous devez les répartir comme il convient.

```
innodb_io_capacity = 2500
innodb_io_capacity_max = 3000
```

## Utilisation de la RAM

Définissez la RAM disponible pour MySQL. Bien que la règle de base soit 70-80%, cela dépend vraiment de la question de savoir si votre instance est dédiée ou non à MySQL et de la quantité de mémoire vive disponible. Ne *gaspillez pas la* RAM (c.-à-d. Les ressources) si vous en avez beaucoup.

```
innodb_buffer_pool_size = 10G
```

## Sécurisation du cryptage MySQL

Le chiffrement par défaut `aes-128-ecb` utilise le mode ECB (Electronic Codebook), qui n'est pas sécurisé et ne devrait jamais être utilisé. Ajoutez plutôt ce qui suit à votre fichier de configuration:

```
block_encryption_mode = aes-256-cbc
```

Lire Configuration et réglage en ligne: <https://riptutorial.com/fr/mysql/topic/3134/configuration-et-reglage>

# Chapitre 12: Connexion avec UTF-8 en utilisant divers langages de programmation.

## Exemples

### Python

1ère ou 2ème ligne dans le code source (pour avoir des littéraux dans le code codé en utf8):

```
# -*- coding: utf-8 -*-
```

Connexion:

```
db = MySQLdb.connect(host=DB_HOST, user=DB_USER, passwd=DB_PASS, db=DB_NAME,
                    charset="utf8mb4", use_unicode=True)
```

Pour les pages Web, l'une d'entre elles:

```
<meta charset="utf-8" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

### PHP

Dans php.ini (c'est la valeur par défaut après PHP 5.6):

```
default_charset UTF-8
```

Lors de la création d'une page Web:

```
header('Content-type: text/plain; charset=UTF-8');
```

Lors de la connexion à MySQL:

```
(for mysql:) Do not use the mysql_* API!
(for mysqli:) $mysqli_obj->set_charset('utf8mb4');
(for PDO:) $db = new PDO('dblib:host=host;dbname=db;charset=utf8', $user, $pwd);
```

Dans le code, n'utilisez aucune routine de conversion.

Pour la saisie de données,

```
<form accept-charset="UTF-8">
```

Pour JSON, évitez `\uxxxx` :

```
$t = json_encode($s, JSON_UNESCAPED_UNICODE);
```

Lire Connexion avec UTF-8 en utilisant divers langages de programmation. en ligne:  
<https://riptutorial.com/fr/mysql/topic/7332/connexion-avec-utf-8-en-utilisant-divers-langages-de-programmation->

# Chapitre 13: Conversion de MyISAM à InnoDB

## Exemples

### Conversion de base

```
ALTER TABLE foo ENGINE=InnoDB;
```

Cela convertit la table, mais ne prend pas en compte les différences entre les moteurs. La plupart des différences n'auront aucune importance, en particulier pour les petites tables. Mais pour les tables plus fréquentées, d'autres considérations doivent être prises en compte. [Considérations de conversion](#)

### Conversion de toutes les tables dans une base de données

Pour convertir facilement toutes les tables d'une base de données, utilisez ce qui suit:

```
SET @DB_NAME = DATABASE();

SELECT CONCAT('ALTER TABLE `', table_name, '` ENGINE=InnoDB;') AS sql_statements
FROM information_schema.tables
WHERE table_schema = @DB_NAME
AND `ENGINE` = 'MyISAM'
AND `TABLE_TYPE` = 'BASE TABLE';
```

**REMARQUE:** Vous devez être connecté à votre base de données pour que la fonction `DATABASE()` fonctionne, sinon elle renverra la `NULL`. Cela s'applique principalement au client mysql standard livré avec le serveur car il permet de se connecter sans spécifier de base de données.

Exécutez cette instruction SQL pour récupérer toutes les tables `MyISAM` de votre base de données.

Enfin, copiez le résultat et exécutez des requêtes SQL à partir de celui-ci.

Lire [Conversion de MyISAM à InnoDB en ligne](#):

<https://riptutorial.com/fr/mysql/topic/3135/conversion-de-myisam-a-innodb>



# Chapitre 14: Création de bases de données

## Syntaxe

- `CREATE {BASE DE DONNEES | SCHEMA} [SI N'EXISTE PAS] db_name [create_specification] ///` Pour créer une base de données
- `DROP {BASE DE DONNEES | SCHEMA} [IF EXISTS] nom_base ///` Pour supprimer une base de données

## Paramètres

| Paramètre                         | Détails   |
|-----------------------------------|---|
| <code>CREATE DATABASE</code>      | Crée une base de données avec le nom donné  |
| <code>CRÉER UN SCHEMA</code>      | Ceci est un synonyme de <code>CREATE DATABASE</code>  |
| <code>SI PAS EXISTANT</code>      | Utilisé pour éviter les erreurs d'exécution, si la base de données spécifiée existe déjà  |
| <code>create_specification</code> | <code>create_specification options create_specification</code> spécifient les caractéristiques de la base de données, telles que <code>CHARACTER SET</code> et <code>COLLATE</code> (classement de la base de données). |

## Exemples

### Créer une base de données, des utilisateurs et des subventions

Créez une base de données. Notez que le mot abrégé `SCHEMA` peut être utilisé comme synonyme.

```
CREATE DATABASE Baseball; -- creates a database named Baseball
```

Si la base de données existe déjà, l'erreur 1007 est renvoyée. Pour contourner cette erreur, essayez:

```
CREATE DATABASE IF NOT EXISTS Baseball;
```

De même,

```
DROP DATABASE IF EXISTS Baseball; -- Drops a database if it exists, avoids Error 1008
DROP DATABASE xyz; -- If xyz does not exist, ERROR 1008 will occur
```

En raison des possibilités d'erreur ci-dessus, les instructions DDL sont souvent utilisées avec `IF EXISTS`.

On peut créer une base de données avec un jeu de caractères et un classement par défaut. Par exemple:

```
CREATE DATABASE Baseball CHARACTER SET utf8 COLLATE utf8_general_ci;

SHOW CREATE DATABASE Baseball;
+-----+-----+
| Database | Create Database |
+-----+-----+
| Baseball | CREATE DATABASE `Baseball` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+-----+-----+
```

Voir vos bases de données actuelles:

```
SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| ajax_stuff |
| Baseball |
+-----+
```

Définissez la base de données active et consultez des informations:

```
USE Baseball; -- set it as the current database
SELECT @@character_set_database as cset, @@collation_database as col;
+-----+-----+
| cset | col |
+-----+-----+
| utf8 | utf8_general_ci |
+-----+-----+
```

Ce qui précède montre le groupe de caractères et le classement par défaut pour la base de données.

Créez un utilisateur:

```
CREATE USER 'John123'@'%' IDENTIFIED BY 'OpenSesame';
```

Ce qui précède crée un utilisateur John123, capable de se connecter avec n'importe quel nom d'hôte en raison du caractère générique `%`. Le mot de passe de l'utilisateur est défini sur "OpenSesame" qui est haché.

Et créer un autre:

```
CREATE USER 'John456'@'%' IDENTIFIED BY 'somePassword';
```

Montrer que les utilisateurs ont été créés en examinant la base de données spéciale `mysql`:

```
SELECT user,host,password from mysql.user where user in ('John123','John456');
+-----+-----+-----+
| user   | host | password |
+-----+-----+-----+
| John123 | %    | *E6531C342ED87 ..... |
| John456 | %    | *B04E11FAAAE9A ..... |
+-----+-----+-----+
```

Notez qu'à ce stade, les utilisateurs ont été créés, mais sans aucune autorisation d'utiliser la base de données Baseball.

Travailler avec des autorisations pour les utilisateurs et les bases de données. Accordez des droits à l'utilisateur John123 pour disposer de tous les privilèges sur la base de données Baseball, et ne sélectionnez que les droits SELECT pour l'autre utilisateur:

```
GRANT ALL ON Baseball.* TO 'John123'@'%';
GRANT SELECT ON Baseball.* TO 'John456'@'%';
```

Vérifiez ce qui précède:

```
SHOW GRANTS FOR 'John123'@'%';
+-----+
-----+
| Grants for John123@%
|
+-----+
-----+
| GRANT USAGE ON *.* TO 'John123'@%' IDENTIFIED BY PASSWORD '*E6531C342ED87
..... |
| GRANT ALL PRIVILEGES ON `baseball`.* TO 'John123'@%'
|
+-----+
-----+

SHOW GRANTS FOR 'John456'@'%';
+-----+
-----+
| Grants for John456@%
|
+-----+
-----+
| GRANT USAGE ON *.* TO 'John456'@%' IDENTIFIED BY PASSWORD '*B04E11FAAAE9A
..... |
| GRANT SELECT ON `baseball`.* TO 'John456'@%'
|
+-----+
-----+
```

Notez que l'utilisation de la `GRANT USAGE` que vous verrez toujours signifie simplement que l'utilisateur peut se connecter. C'est tout ce que cela signifie.

## MyDatabase

Vous devez créer votre propre base de données et ne pas utiliser l'écriture dans les bases de données existantes. Ce sera probablement l'une des toutes premières choses à faire après la

première connexion.

```
CREATE DATABASE my_db;
USE my_db;
CREATE TABLE some_table;
INSERT INTO some_table ...;
```

Vous pouvez référencer votre table en vous qualifiant avec le nom de la base de données:

```
my_db.some_table .
```

## Bases de données système

Les bases de données suivantes existent pour l'utilisation de MySQL. Vous pouvez les lire ( `SELECT` ), mais vous ne devez pas écrire ( `INSERT / UPDATE / DELETE` ) les tableaux qu'ils contiennent. (Il y a quelques exceptions.)

- `mysql` - repository pour les informations `GRANT` et quelques autres choses.
- `information_schema` - Les tables ici sont «virtuelles» en ce sens qu'elles se manifestent en réalité par des structures en mémoire. Leur contenu inclut le schéma de toutes les tables.
- `performance_schema` - ?? [veuillez accepter, puis éditer]
- autres?? (pour MariaDB, Galera, TokuDB, etc.)

## Création et sélection d'une base de données

Si l'administrateur crée votre base de données pour vous lors de la configuration de vos autorisations, vous pouvez commencer à l'utiliser. Sinon, vous devez le créer vous-même:

```
mysql> CREATE DATABASE menagerie;
```

Sous Unix, les noms de base de données sont sensibles à la casse (contrairement aux mots-clés SQL), vous devez donc toujours faire référence à votre base de données en tant que `menagerie`, et non `Menagerie`, `MENAGERIE` ou toute autre variante. Cela est également vrai pour les noms de table. (Sous Windows, cette restriction ne s'applique pas, même si vous devez faire référence à des bases de données et à des tables utilisant la même lettre dans une requête donnée. Cependant, pour diverses raisons, la meilleure pratique recommandée consiste à la base de données a été créée.)

La création d'une base de données ne la sélectionne pas pour être utilisée. vous devez le faire explicitement. Pour faire de la `menagerie` la base de données courante, utilisez cette instruction:

```
mysql> USE menagerie
Database changed
```

Votre base de données doit être créée une seule fois, mais vous devez la sélectionner pour l'utiliser chaque fois que vous démarrez une session `mysql`. Vous pouvez le faire en émettant une instruction `USE` comme indiqué dans l'exemple. Vous pouvez également sélectionner la base de données sur la ligne de commande lorsque vous appelez `mysql`. Indiquez simplement son nom après tout paramètre de connexion que vous pourriez avoir à fournir. Par exemple:

```
shell> mysql -h host -u user -p menagerie  
Enter password: *****
```

Lire Création de bases de données en ligne: <https://riptutorial.com/fr/mysql/topic/600/creation-de-bases-de-donnees>

# Chapitre 15: Création de table

## Syntaxe

- `CREATE TABLE nom_table (nom_colonne1 type_données (taille), nom_colonne2 type_données (taille), nom_colonne3 type_données (taille), ...);` // Création de table de base
- `CREATE TABLE nom_table [IF NOT EXISTS] (nom_colonne1 type_données (taille), nom_colonne2 type_données (taille), nom_colonne3 type_données (taille), ...);` // Création de la table en vérifiant existante
- `CREATE [TEMPORARY] TABLE nom_table [IF NOT EXISTS] (nom_colonne1 type_données (taille), nom_colonne2 type_données (taille), nom_colonne3 type_données (taille), ...);` // Création de table temporaire
- `CREATE TABLE new_tbl [AS] SELECT * FROM orig_tbl;` // Création de table à partir de SELECT

## Remarques

L'instruction `CREATE TABLE` doit se terminer par une spécification `ENGINE` :

```
CREATE TABLE table_name ( column_definitions ) ENGINE=engine;
```

### Certaines options sont les suivantes:

- `InnoDB` : (Par défaut depuis la version 5.5.5) C'est un moteur sans danger pour la transition (compatible ACID). Il dispose de capacités de validation et de restauration des transactions, de récupération après incident et de verrouillage au niveau des lignes.
- `MyISAM` : (Valeur par défaut avant la version 5.5.5) C'est un moteur simple à utiliser. Il ne prend pas en charge les transactions, ni les clés étrangères, mais il est utile pour l'entreposage de données.
- `Memory` : stocke toutes les données dans la RAM pour des opérations extrêmement rapides, mais la date de la table sera perdue lors du redémarrage de la base de données.

Plus d'options de moteur [ici](#) .

## Exemples

### Création de table de base

L'instruction `CREATE TABLE` est utilisée pour créer une table dans une base de données MySQL.

```
CREATE TABLE Person (  
  `PersonID`          INTEGER NOT NULL PRIMARY KEY,
```

```
`LastName`      VARCHAR(80),
`FirstName`     VARCHAR(80),
`Address`       TEXT,
`City`          VARCHAR(100)
) Engine=InnoDB;
```

Chaque définition de champ doit avoir:

1. Nom de champ: Nom de champ valide. Assurez-vous de bien inscrire les noms dans ` -chars. Cela garantit que vous pouvez utiliser par exemple des caractères spatiaux dans le nom du champ.
2. Type de données [Longueur]: si le champ est `CHAR` ou `VARCHAR` , il est obligatoire de spécifier une longueur de champ.
3. Attributs `NULL` | `NOT NULL` : Si `NOT NULL` est spécifié, toute tentative de stockage d'une valeur `NULL` dans ce champ échouera.
4. Voir plus sur les types de données et leurs attributs [ici](#) .

`Engine=...` est un paramètre facultatif utilisé pour spécifier le moteur de stockage de la table. Si aucun moteur de stockage n'est spécifié, la table sera créée à l'aide du moteur de stockage de table par défaut du serveur (généralement InnoDB ou MyISAM).

## Définition des valeurs par défaut

De plus, là où il est logique, vous pouvez définir une valeur par défaut pour chaque champ en utilisant `DEFAULT` :

```
CREATE TABLE Address (
  `AddressID`      INTEGER NOT NULL PRIMARY KEY,
  `Street`         VARCHAR(80),
  `City`           VARCHAR(80),
  `Country`        VARCHAR(80) DEFAULT "United States",
  `Active`         BOOLEAN DEFAULT 1,
) Engine=InnoDB;
```

Si pendant l'insertion, aucune `Street` n'est spécifiée, ce champ sera `NULL` lors de sa récupération. Lorsqu'aucun `Country` n'est spécifié lors de l'insertion, il sera par défaut "États-Unis".

Vous pouvez définir des valeurs par défaut pour tous les types de colonne, à l' [exception](#) des champs `BLOB` , `TEXT` , `GEOMETRY` et `JSON` .

## Création de table avec clé primaire

```
CREATE TABLE Person (
  PersonID        INT UNSIGNED NOT NULL,
  LastName         VARCHAR(66) NOT NULL,
  FirstName        VARCHAR(66),
  Address          VARCHAR(255),
  City             VARCHAR(66),
  PRIMARY KEY (PersonID)
);
```

Une **clé primaire** est un identificateur unique ou multi-colonnes `NOT NULL` qui identifie de manière unique une ligne d'une table. Un `index` est créé et, s'il n'est pas déclaré explicitement comme `NOT NULL`, MySQL les déclarera de manière implicite et silencieuse.

Une table ne peut avoir qu'une seule `PRIMARY KEY` et chaque table est recommandée. InnoDB en créera automatiquement un en son absence (comme vu dans la [documentation MySQL](#)), bien que cela soit moins souhaitable.

Souvent, une `INT AUTO_INCREMENT` également connue sous le nom de "clé de substitution" est utilisée pour l'optimisation d'index léger et les relations avec d'autres tables. Cette valeur augmente (normalement) de 1 à chaque fois qu'un nouvel enregistrement est ajouté, à partir d'une valeur par défaut de 1.

Cependant, malgré son nom, le but n'est pas de garantir que les valeurs sont incrémentielles, mais simplement qu'elles sont séquentielles et uniques.

Une valeur `INT` incrémentée automatiquement ne sera pas réinitialisée à sa valeur de démarrage par défaut si toutes les lignes de la table sont supprimées, à moins que la table ne soit tronquée à l'aide de l' `TRUNCATE TABLE`.

---

## Définition d'une colonne en tant que clé primaire (définition en ligne)

Si la clé primaire est constituée d'une seule colonne, la clause `PRIMARY KEY` peut être intégrée à la définition de la colonne:

```
CREATE TABLE Person (  
    PersonID      INT UNSIGNED NOT NULL PRIMARY KEY,  
    LastName     VARCHAR(66) NOT NULL,  
    FirstName    VARCHAR(66),  
    Address      VARCHAR(255),  
    City         VARCHAR(66)  
);
```

Cette forme de la commande est plus courte et plus facile à lire.

---

## Définition d'une clé primaire à plusieurs colonnes

Il est également possible de définir une clé primaire comprenant plusieurs colonnes. Cela peut être fait par exemple sur la table enfant d'une relation de clé étrangère. Une clé primaire multi-colonnes est définie en répertoriant les colonnes participantes dans une clause `PRIMARY KEY` distincte. La syntaxe inline n'est pas autorisée ici, car une seule colonne peut être déclarée en ligne `PRIMARY KEY`. Par exemple:



```
CREATE TABLE invoice_line_items (
  LineNum      SMALLINT UNSIGNED NOT NULL,
  InvoiceNum    INT UNSIGNED NOT NULL,
  -- Other columns go here
  PRIMARY KEY (InvoiceNum, LineNum),
  FOREIGN KEY (InvoiceNum) REFERENCES -- references to an attribute of a table
);
```

Notez que les colonnes de la clé primaire *doivent* être spécifiées dans l'ordre de tri logique, qui *peut* être différent de l'ordre dans lequel les colonnes ont été définies, comme dans l'exemple ci-dessus.

Les index plus grands nécessitent plus d'espace disque, de mémoire et d'E / S. Les clés doivent donc être aussi petites que possible (en particulier en ce qui concerne les touches composées). Dans InnoDB, chaque «index secondaire» inclut une copie des colonnes de la `PRIMARY KEY`.

## Création de table avec clé étrangère

```
CREATE TABLE Account (
  AccountID    INT UNSIGNED NOT NULL,
  AccountNo    INT UNSIGNED NOT NULL,
  PersonID     INT UNSIGNED,
  PRIMARY KEY (AccountID),
  FOREIGN KEY (PersonID) REFERENCES Person (PersonID)
) ENGINE=InnoDB;
```

**Clé étrangère:** une clé étrangère ( `FK` ) est soit une colonne unique, soit un composite de colonnes à colonnes multiples, dans une table de *référence*. Ce `FK` est confirmé pour exister dans la table *référéncée*. Il est fortement recommandé que la clé de table *référéncée* confirmant le `FK` soit une clé primaire, mais cela n'est pas appliqué. Il est utilisé comme une recherche rapide dans les *références* où il n'a pas besoin d'être unique et peut en fait être un index le plus à gauche.

Les relations de clé étrangère impliquent une table parent contenant les valeurs de données centrales et une table enfant avec des valeurs identiques renvoyant à son parent. La clause `FOREIGN KEY` est spécifiée dans la table enfant. Les tables parent et enfant doivent utiliser le même moteur de stockage. Ils ne doivent pas être des tables **TEMPORAIRES**.

Les colonnes correspondantes dans la clé étrangère et la clé référencée doivent avoir des types de données similaires. La taille et le signe des types entiers doivent être identiques. La longueur des types de chaîne ne doit pas nécessairement être la même. Pour les colonnes de chaîne non binaires (caractères), le jeu de caractères et le classement doivent être identiques.

**Remarque:** les contraintes de clé étrangère sont prises en charge sous le moteur de stockage InnoDB (pas MyISAM ou MEMORY). Les configurations de base de données utilisant d'autres moteurs accepteront cette instruction `CREATE TABLE` mais ne respecteront pas les contraintes de clé étrangère. (Bien que les nouvelles versions de MySQL soient par défaut à `InnoDB`, mais il est recommandé d'être explicite.)

## Cloner une table existante

Un tableau peut être répliqué comme suit:

```
CREATE TABLE ClonedPersons LIKE Persons;
```

La nouvelle table aura exactement la même structure que la table d'origine, y compris les index et les attributs de colonne.

Outre la création manuelle d'une table, il est également possible de créer une table en sélectionnant des données d'une autre table:

```
CREATE TABLE ClonedPersons SELECT * FROM Persons;
```

Vous pouvez utiliser l'une des fonctionnalités normales d'une `SELECT` pour modifier les données au fur et à mesure:

```
CREATE TABLE ModifiedPersons
SELECT PersonID, FirstName + LastName AS FullName FROM Persons
WHERE LastName IS NOT NULL;
```

Les clés primaires et les index ne seront pas conservés lors de la création de tables à partir de `SELECT`. Vous devez les redéclarer:

```
CREATE TABLE ModifiedPersons (PRIMARY KEY (PersonID))
SELECT PersonID, FirstName + LastName AS FullName FROM Persons
WHERE LastName IS NOT NULL;
```

## CREATE TABLE FROM SELECT

Vous pouvez créer une table à partir d'une autre en ajoutant une `SELECT` à la fin de l'instruction

`CREATE TABLE` :

```
CREATE TABLE stack (
    id_user INT,
    username VARCHAR(30),
    password VARCHAR(30)
);
```

**Créez une table dans la même base de données:**

```
-- create a table from another table in the same database with all attributes
CREATE TABLE stack2 AS SELECT * FROM stack;

-- create a table from another table in the same database with some attributes
CREATE TABLE stack3 AS SELECT username, password FROM stack;
```

**Créez des tables à partir de différentes bases de données:**

```
-- create a table from another table from another database with all attributes
CREATE TABLE stack2 AS SELECT * FROM second_db.stack;
```

```
-- create a table from another table from another database with some attributes
CREATE TABLE stack3 AS SELECT username, password FROM second_db.stack;
```

## NB

Pour créer une table identique à une autre table existant dans une autre base de données, vous devez spécifier le nom de la base de données comme suit:

```
FROM NAME_DATABASE.name_table
```

## Afficher la structure du tableau

Si vous souhaitez voir les informations de schéma de votre table, vous pouvez utiliser l'une des options suivantes:

```
SHOW CREATE TABLE child; -- Option 1

CREATE TABLE `child` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `fullName` varchar(100) NOT NULL,
  `myParent` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `mommy_daddy` (`myParent`),
  CONSTRAINT `mommy_daddy` FOREIGN KEY (`myParent`) REFERENCES `parent` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

S'il est utilisé à partir de l'outil en ligne de commande mysql, c'est moins prolix:

```
SHOW CREATE TABLE child \G
```

Une manière moins descriptive de montrer la structure de la table:

```
mysql> CREATE TABLE Tab1(id int, name varchar(30));
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> DESCRIBE Tab1; -- Option 2
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)       | YES  |     | NULL    |       |
| name  | varchar(30)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
```

**DESCRIBE** et **DESC** donnent le même résultat.

Pour voir `DESCRIBE` exécuté sur toutes les tables d'une base de données à la fois, voir cet [exemple](#)

**Table Create With TimeStamp Column pour afficher la dernière mise à jour**

La colonne **TIMESTAMP** affichera la dernière mise à jour de la ligne.

```
CREATE TABLE `TestLastUpdate` (  
  `ID` INT NULL,  
  `Name` VARCHAR(50) NULL,  
  `Address` VARCHAR(50) NULL,  
  `LastUpdate` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP  
)  
COMMENT='Last Update'  
;
```

Lire Création de table en ligne: <https://riptutorial.com/fr/mysql/topic/795/creation-de-table>

---

# Chapitre 16: Créer un nouvel utilisateur

## Remarques

Pour afficher une liste d'utilisateurs MySQL, nous utilisons la commande suivante:

```
SELECT User,Host FROM mysql.user;
```

## Exemples

### Créer un utilisateur MySQL

Pour créer un nouvel utilisateur, nous devons suivre des étapes simples comme ci-dessous:

**Étape 1:** Connectez - vous à MySQL tant que `root`

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

**Étape 2:** Nous verrons l'invite de commande `mysql`

```
mysql> CREATE USER 'my_new_user'@'localhost' IDENTIFIED BY 'test_password';
```

Ici, nous avons créé avec succès un nouvel utilisateur, mais cet utilisateur n'aura aucune permissions , donc pour attribuer des permissions à l'utilisateur utilisez la commande suivante:

```
mysql> GRANT ALL PRIVILEGES ON my_db.* TO 'my_new_user'@'localhost' identified by 'my_password';
```

### Spécifiez le mot de passe

L'utilisation de base est la suivante:

```
mysql> CREATE USER 'my_new_user'@'localhost' IDENTIFIED BY 'test_password';
```

Cependant, dans les situations où il est déconseillé de coder en dur le mot de passe en clair, il est également possible de spécifier directement, à l'aide de la directive `PASSWORD` , la valeur hachée renvoyée par la fonction `PASSWORD()` :

```
mysql> select PASSWORD('test_password'); -- returns *4414E26EDED6D661B5386813EBBA95065DBC4728
mysql> CREATE USER 'my_new_user'@'localhost' IDENTIFIED BY PASSWORD
 '*4414E26EDED6D661B5386813EBBA95065DBC4728';
```

### Créer un nouvel utilisateur et accorder tous les privilèges au schéma

```
grant all privileges on schema_name.* to 'new_user_name'@'%' identified by 'newpassword';
```

Attention: Ceci peut être utilisé pour créer un nouvel utilisateur root

## Renommer l'utilisateur

```
rename user 'user'@'%' to 'new_name'@'%';
```

Si vous créez un utilisateur par erreur, vous pouvez changer son nom

Lire **Créer un nouvel utilisateur en ligne**: <https://riptutorial.com/fr/mysql/topic/3508/creer-un-nouvel-utilisateur>

# Chapitre 17: EFFACER

## Syntaxe

- DELETE [LOW\_PRIORITY] [QUICK] [IGNORE] FROM table [conditions WHERE] [expression ORDER BY [ASC | DESC]] [LIMIT Number\_rows]; /// Syntaxe pour supprimer les lignes de la table unique

## Paramètres

| Paramètre               | Détails   |
|-------------------------|---|
| PRIORITÉ BASSE          | Si <code>LOW_PRIORITY</code> est fourni, la suppression sera retardée jusqu'à ce qu'il n'y ait plus de processus à lire dans la table.                            |
| IGNORER                 | Si <code>IGNORE</code> est fourni, toutes les erreurs rencontrées lors de la suppression sont ignorées  |
| table                   | La table à partir de laquelle vous allez supprimer des enregistrements  |
| O conditions conditions | Les conditions à remplir pour que les enregistrements soient supprimés. Si aucune condition n'est fournie, tous les enregistrements de la table seront supprimés. |
| Expression ORDER BY     | Si <code>ORDER BY</code> est fourni, les enregistrements seront supprimés dans l'ordre indiqué  |
| LIMITE                  | Il contrôle le nombre maximal d'enregistrements à supprimer de la table. Les <code>number_rows</code> donnés seront supprimés.                                    |

## Exemples

### Supprimer avec la clause Where

```
DELETE FROM `table_name` WHERE `field_one` = 'value_one'
```

Cela supprimera toutes les lignes de la table où le contenu de la zone `field_one` à cette ligne correspond à 'value\_one'.

La clause `WHERE` fonctionne de la même manière qu'une sélection, de sorte que des éléments tels que `>`, `<`, `<>` ou `LIKE` peuvent être utilisés.

**Remarque:** il est nécessaire d'utiliser des clauses conditionnelles (`WHERE`, `LIKE`) dans la requête de suppression. Si vous n'utilisez aucune clause conditionnelle, toutes les données de

cette table seront supprimées.

## Supprimer toutes les lignes d'une table

```
DELETE FROM table_name ;
```

Cela supprimera tout, toutes les lignes de la table. C'est l'exemple le plus fondamental de la syntaxe. Il montre également que les instructions `DELETE` doivent vraiment être utilisées avec précaution car elles peuvent vider une table si la clause `WHERE` est omise.

## Supprimer les suppressions

```
DELETE FROM `table_name` WHERE `field_one` = 'value_one' LIMIT 1
```

Cela fonctionne de la même manière que dans l'exemple de la clause «Delete with Where», mais elle arrêtera la suppression une fois que le nombre limité de lignes aura été supprimé.

Si vous limitez les lignes à supprimer de la sorte, sachez que cela supprimera la première ligne correspondant aux critères. Ce n'est peut-être pas ce à quoi vous vous attendez, car les résultats peuvent être non triés s'ils ne sont pas explicitement commandés.

## Suppressions multi-tables

L'instruction `DELETE` de MySQL peut utiliser la construction `JOIN`, permettant également de spécifier les tables à supprimer. Ceci est utile pour éviter les requêtes imbriquées. Compte tenu du schéma:

```
create table people
(
  id int primary key,
  name varchar(100) not null,
  gender char(1) not null
);
insert people (id,name,gender) values
(1,'Kathy','f'), (2,'John','m'), (3,'Paul','m'), (4,'Kim','f');

create table pets
(
  id int auto_increment primary key,
  ownerId int not null,
  name varchar(100) not null,
  color varchar(100) not null
);
insert pets(ownerId,name,color) values
(1,'Rover','beige'), (2,'Bubbles','purple'), (3,'Spot','black and white'),
(1,'Rover2','white');
```

| id | prénom | le genre |
|----|--------|----------|
| 1  | Kathy  | F        |
| 2  | John   | m        |



| id | prénom | le genre |
|----|--------|----------|
| 3  | Paul   | m        |
| 4  | Kim    | F        |

| id | ownerId | prénom   | Couleur |
|----|---------|----------|---------|
| 1  | 1       | Vagabond | beige   |
| 2  | 2       | Bulles   | violet  |
| 4  | 1       | Rover2   | blanc   |

Si nous voulons supprimer les animaux de compagnie de Paul, la déclaration

```
DELETE p2
FROM pets p2
WHERE p2.ownerId in (
  SELECT p1.id
  FROM people p1
  WHERE p1.name = 'Paul');
```

peut être réécrit comme suit:

```
DELETE p2 -- remove only rows from pets
FROM people p1
JOIN pets p2
ON p2.ownerId = p1.id
WHERE p1.name = 'Paul';
```

### *1 rangée supprimée*

Spot est supprimé de Pets

`p1` et `p2` sont des alias pour les noms de tables, particulièrement utiles pour les noms de tables longues et la facilité de lecture.

Pour retirer à la fois la personne et l'animal de compagnie:

```
DELETE p1, p2 -- remove rows from both tables
FROM people p1
JOIN pets p2
ON p2.ownerId = p1.id
WHERE p1.name = 'Paul';
```

### *2 lignes supprimées*

Spot est supprimé de Pets

Paul est supprimé de People

## clés étrangères

Lorsque l'instruction DELETE appelle des tables avec une clé externe, l'optimiseur peut traiter les tables dans un ordre qui ne suit pas la relation. Ajouter par exemple une clé étrangère à la définition des `pets` de `pets`

```
ALTER TABLE pets ADD CONSTRAINT `fk_pets_2_people` FOREIGN KEY (ownerId) references people(id)
ON DELETE CASCADE;
```

le moteur peut essayer de supprimer les entrées des `people` avant les `pets`, provoquant ainsi l'erreur suivante:

```
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
(`test`.`pets`, CONSTRAINT `pets_ibfk_1` FOREIGN KEY (`ownerId`) REFERENCES `people` (`id`))
```

La solution dans ce cas est de supprimer la ligne des `people` et de s'appuyer sur les capacités `ON DELETE InnoDB` pour propager la suppression:

```
DELETE FROM people
WHERE name = 'Paul';
```

### 2 lignes supprimées

Paul est supprimé de People

Spot est supprimé en cascade de Pets

Une autre solution consiste à désactiver temporairement le contrôle sur les clés étrangères:

```
SET foreign_key_checks = 0;
DELETE p1, p2 FROM people p1 JOIN pets p2 ON p2.ownerId = p1.id WHERE p1.name = 'Paul';
SET foreign_key_checks = 1;
```

## Suppression de base

```
DELETE FROM `myTable` WHERE `someColumn` = 'something'
```

La clause `WHERE` est facultative mais sans elle, toutes les lignes sont supprimées.

## SUPPRIMER vs TRUNCATE

```
TRUNCATE tableName;
```

Cela **supprimera** toutes les données et réinitialisera l'index `AUTO_INCREMENT`. C'est beaucoup plus rapide que `DELETE FROM tableName` sur un énorme ensemble de données. Cela peut être très utile lors du développement / test.

Lorsque vous **tronquez** une table, le serveur SQL ne supprime pas les données, il supprime la table et la recrée, libérant ainsi les pages, ce qui permet de récupérer les données tronquées avant les pages écrasées. (L'espace ne peut pas être immédiatement récupéré pour `innodb_file_per_table=OFF`.)

## Multi-table DELETE

MySQL permet de spécifier à partir de quelle table les lignes correspondantes doivent être supprimées

```
-- remove only the employees
DELETE e
FROM Employees e JOIN Department d ON e.department_id = d.department_id
WHERE d.name = 'Sales'
```

```
-- remove employees and department
DELETE e, d
FROM Employees e JOIN Department d ON e.department_id = d.department_id
WHERE d.name = 'Sales'
```

```
-- remove from all tables (in this case same as previous)
DELETE
FROM Employees e JOIN Department d ON e.department_id = d.department_id
WHERE d.name = 'Sales'
```

Lire EFFACER en ligne: <https://riptutorial.com/fr/mysql/topic/1487/effacer>

# Chapitre 18: ENUM

## Exemples

### Pourquoi ENUM?

ENUM fournit un moyen de fournir un attribut pour une ligne. Les attributs avec un petit nombre d'options non numériques fonctionnent mieux. Exemples:

```
reply ENUM('yes', 'no')
gender ENUM('male', 'female', 'other', 'decline-to-state')
```

Les valeurs sont des chaînes:

```
INSERT ... VALUES ('yes', 'female')
SELECT ... --> yes female
```

### TINYINT comme alternative

Disons que nous avons

```
type ENUM('fish', 'mammal', 'bird')
```

Une alternative est

```
type TINYINT UNSIGNED
```

plus

```
CREATE TABLE AnimalTypes (
  type TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL COMMENT " ('fish', 'mammal', 'bird') ",
  PRIMARY KEY (type),
  INDEX (name)
) ENGINE=InnoDB
```

ce qui ressemble beaucoup à une table plusieurs-à-plusieurs.

Comparaison, et meilleure ou pire que ENUM:

- (pire) INSERT: besoin de rechercher le `type`
- (pire) SELECT: besoin de JOINDRE pour obtenir la chaîne (ENUM vous donne la chaîne sans effort)
- (mieux) Ajouter de nouveaux types: insérez simplement dans ce tableau. Avec ENUM, vous devez faire un ALTER TABLE.
- (idem) Chaque technique (pour 255 valeurs maximum) ne prend que 1 octet.

- (mixed) Il y a aussi un problème d'intégrité des données: `TINYINT` admettra des valeurs non valides; alors que `ENUM` leur `ENUM` une valeur de chaîne vide spéciale (sauf si le mode SQL strict est activé, auquel cas ils sont rejetés). Une meilleure intégrité des données peut être obtenue avec `TINYINT` en faisant une clé étrangère dans une table de recherche: ce qui, avec les requêtes / jointures appropriées, présente néanmoins un faible coût pour atteindre l'autre table. (Les `FOREIGN KEYS` ne sont pas gratuites.)

## VARCHAR comme alternative

Disons que nous avons

```
type ENUM('fish','mammal','bird')
```

Une alternative est

```
type VARCHAR(20) COMMENT "fish, bird, etc"
```

Ceci est assez ouvert dans la mesure où de nouveaux types sont trivialement ajoutés.

Comparaison, et meilleure ou pire que `ENUM`:

- (même) `INSERT`: fournissez simplement la chaîne
- (pire?) Sur `INSERT` une faute de frappe passera inaperçue
- (même) `SELECT`: la chaîne actuelle est renvoyée
- (pire) Beaucoup plus d'espace est consommé

## Ajouter une nouvelle option

```
ALTER TABLE tbl MODIFY COLUMN type ENUM('fish','mammal','bird','insect');
```

Remarques

- Comme avec tous les cas de `MODIFY COLUMN`, vous devez inclure `NOT NULL` et tous les autres qualificatifs qui existaient à l'origine, sinon ils seront perdus.
- Si vous ajoutez à la fin de la liste et que la liste contient moins de 256 éléments, le programme `ALTER` se fait simplement en modifiant le schéma. C'est-à-dire qu'il n'y aura pas de copie longue de la table. (Les anciennes versions de MySQL n'avaient pas cette optimisation.)

## NULL vs NOT NULL

Des exemples de ce qui se passe lorsque `NULL` et «bad-value» sont stockés dans des colonnes nullable et non nullable. Affiche également l'utilisation de la conversion en numérique via `+0`.

```
CREATE TABLE enum (  
  e      ENUM('yes', 'no')    NOT NULL,  
  enull  ENUM('x', 'y', 'z')  NULL  
);
```

```

INSERT INTO enum (e, enull)
VALUES
  ('yes', 'x'),
  ('no', 'y'),
  (NULL, NULL),
  ('bad-value', 'bad-value');
Query OK, 4 rows affected, 3 warnings (0.00 sec)
Records: 4 Duplicates: 0 Warnings: 3

mysql>SHOW WARNINGS;
+-----+-----+-----+-----+
| Level  | Code | Message                                     |
+-----+-----+-----+-----+
| Warning | 1048 | Column 'e' cannot be null                 |
| Warning | 1265 | Data truncated for column 'e' at row 4    |
| Warning | 1265 | Data truncated for column 'enull' at row 4 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

Ce qui est dans la table après ces insertions. Cela utilise "+0" pour afficher numériquement ce qui est stocké.

```

mysql>SELECT e, e+0 FROM enum;
+-----+-----+
| e   | e+0 |
+-----+-----+
| yes | 1   |
| no  | 2   |
|     | 0   | -- NULL
|     | 0   | -- 'bad-value'
+-----+-----+
4 rows in set (0.00 sec)

mysql>SELECT enull, enull+0 FROM enum;
+-----+-----+
| enull | enull+0 |
+-----+-----+
| x     | 1       |
| y     | 2       |
| NULL  | NULL    |
|       | 0       | -- 'bad-value'
+-----+-----+
4 rows in set (0.00 sec)

```

Lire ENUM en ligne: <https://riptutorial.com/fr/mysql/topic/4425/enum>

---

# Chapitre 19: Erreur 1055: ONLY\_FULL\_GROUP\_BY: quelque chose n'est pas dans la clause GROUP BY ...

## Introduction

Récemment, les nouvelles versions des serveurs MySQL ont commencé à générer des erreurs 1055 pour les requêtes qui fonctionnaient auparavant. Cette rubrique explique ces erreurs. L'équipe MySQL s'est efforcée de retirer l'extension non standard de `GROUP BY`, ou du moins de la rendre plus difficile pour les développeurs d'écriture de requêtes.

## Remarques

MySQL contient depuis longtemps une extension notoire non standard de `GROUP BY`, qui permet un comportement désordonné au nom de l'efficacité. Cette extension a permis à de nombreux développeurs du monde entier d'utiliser `GROUP BY` dans le code de production sans comprendre complètement ce qu'ils faisaient.

En particulier, il est déconseillé d'utiliser `SELECT *` dans une requête `GROUP BY`, car une clause `GROUP BY` standard nécessite l'énumération des colonnes. Beaucoup de développeurs ont malheureusement fait cela.

Lis ça. <https://dev.mysql.com/doc/refman/5.7/en/group-by-handling.html>

L'équipe MySQL a essayé de résoudre ce problème sans perturber le code de production. Ils ont ajouté un indicateur `sql_mode` dans 5.7.5 nommé `ONLY_FULL_GROUP_BY` pour forcer le comportement standard. Dans une version récente, ils ont activé cet indicateur par défaut. Lorsque vous avez mis à niveau votre MySQL local vers la version 5.7.14, l'indicateur a été activé et votre code de production, dépendant de l'ancienne extension, a cessé de fonctionner.

Si vous avez récemment commencé à recevoir 1055 erreurs, quels sont vos choix?

1. Corrigez les requêtes SQL incriminées ou demandez à leurs auteurs de le faire.
2. Revenez à une version de MySQL compatible prête à l'emploi avec le logiciel d'application que vous utilisez.
3. changez le `sql_mode` votre serveur pour vous débarrasser du mode `ONLY_FULL_GROUP_BY` nouvellement défini.

Vous pouvez changer le mode en effectuant une commande `SET`.

```
SET sql_mode =  
'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_EN
```

devrait faire l'affaire si vous le faites juste après que votre application se connecte à MySQL.

Vous pouvez également trouver [le fichier init dans votre installation MySQL](#) , localiser la ligne `sql_mode=` et la modifier pour omettre `ONLY_FULL_GROUP_BY` , puis redémarrer votre serveur.

## Exemples

### Utiliser et utiliser GROUP BY

```
SELECT item.item_id, item.name,      /* not SQL-92 */
       COUNT(*) number_of_uses
FROM item
JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id
```

affichera les lignes d'une table appelée `item` et affichera le nombre de lignes associées dans une table appelée `uses` . Cela fonctionne bien, mais malheureusement, ce n'est pas la norme SQL-92.

Pourquoi pas? car la clause `SELECT` (et la clause `ORDER BY` ) dans les requêtes `GROUP BY` doivent contenir des colonnes

1. mentionné dans la clause `GROUP BY` , ou
2. fonctions d'agrégation telles que `COUNT()` , `MIN()` , etc.

La clause `SELECT` cet exemple mentionne `item.name` , une colonne qui ne répond à aucun de ces critères. MySQL 5.6 et `ONLY_FULL_GROUP_BY` antérieures rejeteront cette requête si le mode SQL contient `ONLY_FULL_GROUP_BY` .

Cet exemple de requête peut être fait pour se conformer à la norme SQL-92 en modifiant la clause `GROUP BY` , comme ceci.

```
SELECT item.item_id, item.name,
       COUNT(*) number_of_uses
FROM item
JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id, item.name
```

La norme SQL-99 suivante permet à une `SELECT` d'omettre les colonnes non agrégées de la clé de groupe si le SGBD peut prouver une dépendance fonctionnelle entre elles et les colonnes de clé de groupe. Comme `item.name` dépend de manière fonctionnelle de `item.item_id` , l'exemple initial est valide SQL-99. MySQL a obtenu un [prouveur de dépendance fonctionnelle](#) dans la version 5.7. L'exemple d'origine fonctionne sous `ONLY_FULL_GROUP_BY` .

### Abuser de GROUP BY pour obtenir des résultats imprévisibles: la loi de Murphy

```
SELECT item.item_id, uses.category, /* nonstandard */
       COUNT(*) number_of_uses
FROM item
```



```
JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id
```

affichera les lignes d'une table appelée `item` et affichera le nombre de lignes associées dans une table appelée `uses`. Il affichera également la valeur d'une colonne appelée `uses.category`.

Cette requête fonctionne dans MySQL (avant l' `ONLY_FULL_GROUP_BY` indicateur `ONLY_FULL_GROUP_BY` ). Il utilise [l'extension non standard de MySQL pour GROUP BY](#) .

Mais la requête a un problème: si plusieurs lignes de la table `uses` correspondent à la condition `ON` dans la clause `JOIN` , MySQL renvoie la colonne `category` partir d'une seule de ces lignes. Quelle rangée? L'auteur de la requête et l'utilisateur de l'application ne le savent pas à l'avance. D'un point de vue formel, c'est *imprévisible* : MySQL peut retourner n'importe quelle valeur.

*Imprévisible* est comme *aléatoire*, avec une différence significative. On pourrait s'attendre à un choix *aléatoire* de changer de temps en temps. Par conséquent, si un choix était aléatoire, vous pourriez le détecter lors du débogage ou du test. Le résultat *imprévisible* est pire: MySQL renvoie le même résultat à chaque fois que vous utilisez la requête, *jusqu'à ce que ce ne soit pas le cas*. Parfois, c'est une nouvelle version du serveur MySQL qui provoque un résultat différent. Parfois, la table est en train de causer le problème. Ce qui peut mal tourner va mal se passer et quand vous ne vous y attendez pas. C'est ce qu'on appelle [la loi de Murphy](#) .

L'équipe de MySQL s'efforce de rendre l'erreur plus difficile pour les développeurs. Les versions plus récentes de MySQL dans la séquence 5.7 ont un indicateur `sql_mode` appelé `ONLY_FULL_GROUP_BY` . Lorsque cet indicateur est défini, le serveur MySQL renvoie l'erreur 1055 et refuse d'exécuter ce type de requête.

## Mauvaise utilisation de GROUP BY avec SELECT \* et comment y remédier.

Parfois, une requête ressemble à ceci, avec un `*` dans la clause `SELECT` .

```
SELECT item.*,          /* nonstandard */
       COUNT(*) number_of_uses
FROM item
JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id
```

Une telle requête doit être refactorisée pour être conforme à la norme `ONLY_FULL_GROUP_BY` .

Pour ce faire, nous avons besoin d' un sous - requête qui utilise `GROUP BY` correctement pour retourner la `number_of_uses` valeur pour chaque `item_id` . Cette sous-requête est courte et simple, car il suffit de regarder la table des `uses` .

```
SELECT item_id, COUNT(*) number_of_uses
FROM uses
GROUP BY item_id
```

Ensuite, nous pouvons joindre cette sous-requête à la table des `item` .

```

SELECT item.*, usecount.number_of_uses
FROM item
JOIN (
        SELECT item_id, COUNT(*) number_of_uses
        FROM uses
        GROUP BY item_id
    ) usecount ON item.item_id = usecount.item_id

```

Cela permet à la clause `GROUP BY` d'être simple et correcte, et nous permet également d'utiliser le spécificateur `*`.

Note: néanmoins, les développeurs avisés évitent d'utiliser le spécificateur `*` dans tous les cas. Il est généralement préférable de répertorier les colonnes de votre choix dans une requête.

## DE N'IMPORTE QUELLE VALEUR()

```

SELECT item.item_id, ANY_VALUE(uses.tag) tag,
       COUNT(*) number_of_uses
FROM item
JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id

```

affiche les lignes d'une table appelée `item`, le nombre de lignes associées et l'une des valeurs de la table associée appelée `uses`.

Vous pouvez penser à [cette fonction](#) `ANY_VALUE()` comme une sorte de fonction d'agrégat étrange. Au lieu de renvoyer un nombre, une somme ou un maximum, le serveur MySQL doit choisir, arbitrairement, une valeur du groupe en question. C'est un moyen de contourner l'erreur 1055.

Soyez prudent lorsque vous utilisez `ANY_VALUE()` dans les requêtes des applications de production.

Cela devrait vraiment s'appeler `SURPRISE_ME()`. Il renvoie la valeur d'une ligne dans le groupe `GROUP BY`. La ligne qu'il retourne est indéterminée. Cela signifie que tout dépend du serveur MySQL. Formellement, il renvoie une valeur imprévisible.

Le serveur ne choisit pas une valeur aléatoire, c'est pire que cela. Il renvoie la même valeur chaque fois que vous exécutez la requête, jusqu'à ce que ce ne soit pas le cas. Cela peut changer ou non quand une table grandit ou rétrécit, ou quand le serveur a plus ou moins de RAM, ou quand la version du serveur change, ou quand Mars est rétrograde (peu importe ce que cela signifie), ou sans aucune raison.

Tu étais prévenu.

Lire Erreur 1055: `ONLY_FULL_GROUP_BY`: quelque chose n'est pas dans la clause `GROUP BY` ... en ligne: <https://riptutorial.com/fr/mysql/topic/8245/erreur-1055--only-full-group-by--quelque-chose-n-est-pas-dans-la-clause-group-by---->

# Chapitre 20: Événements

## Exemples

### Créer un événement

Mysql a sa fonctionnalité EVENT pour éviter les interactions cron compliquées quand une grande partie de ce que vous planifiez est liée à SQL et moins liée au fichier. Voir la page de manuel [ici](#) . Considérez les événements comme des procédures stockées planifiées pour s'exécuter à intervalles réguliers.

Pour gagner du temps dans le débogage des problèmes liés aux événements, gardez à l'esprit que le gestionnaire d'événements global doit être activé pour traiter les événements.

```
SHOW VARIABLES WHERE variable_name='event_scheduler';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| event_scheduler | OFF   |
+-----+-----+
```

Avec OFF, rien ne se déclenche. Alors allumez-le:

```
SET GLOBAL event_scheduler = ON;
```

## Schéma pour tester

```
create table theMessages
(
  id INT AUTO_INCREMENT PRIMARY KEY,
  userId INT NOT NULL,
  message VARCHAR(255) NOT NULL,
  updateDt DATETIME NOT NULL,
  KEY(updateDt)
);

INSERT theMessages(userId,message,updateDt) VALUES (1,'message 123','2015-08-24 11:10:09');
INSERT theMessages(userId,message,updateDt) VALUES (7,'message 124','2015-08-29');
INSERT theMessages(userId,message,updateDt) VALUES (1,'message 125','2015-09-03 12:00:00');
INSERT theMessages(userId,message,updateDt) VALUES (1,'message 126','2015-09-03 14:00:00');
```

Les insertions ci-dessus sont fournies pour montrer un point de départ. Notez que les 2 événements créés ci-dessous nettoient les lignes.

## Créer 2 événements, 1ère course tous les jours, 2ème parcours toutes les 10 minutes

Ignorer ce qu'ils font réellement (jouer les uns contre les autres). Le point est sur l'INTERVALLE et

## la planification.

```
DROP EVENT IF EXISTS `delete7DayOldMessages`;
DELIMITER $$
CREATE EVENT `delete7DayOldMessages`
  ON SCHEDULE EVERY 1 DAY STARTS '2015-09-01 00:00:00'
  ON COMPLETION PRESERVE
DO BEGIN
  DELETE FROM theMessages
  WHERE datediff(now(),updateDt)>6; -- not terribly exact, yesterday but <24hrs is still 1
day

  -- Other code here

END$$
DELIMITER ;
```

...

```
DROP EVENT IF EXISTS `Every_10_Minutes_Cleanup`;
DELIMITER $$
CREATE EVENT `Every_10_Minutes_Cleanup`
  ON SCHEDULE EVERY 10 MINUTE STARTS '2015-09-01 00:00:00'
  ON COMPLETION PRESERVE
DO BEGIN
  DELETE FROM theMessages
  WHERE TIMESTAMPDIFF(HOUR, updateDt, now())>168; -- messages over 1 week old (168 hours)

  -- Other code here

END$$
DELIMITER ;
```

## Afficher les statuts d'événement (différentes approches)

```
SHOW EVENTS FROM my_db_name; -- List all events by schema name (db name)
SHOW EVENTS;
SHOW EVENTS\G; -- <----- I like this one from mysql> prompt
```

```
***** 1. row *****
      Db: my_db_name
      Name: delete7DayOldMessages
      Definer: root@localhost
      Time zone: SYSTEM
      Type: RECURRING
      Execute at: NULL
      Interval value: 1
      Interval field: DAY
      Starts: 2015-09-01 00:00:00
      Ends: NULL
      Status: ENABLED
      Originator: 1
character_set_client: utf8
collation_connection: utf8_general_ci
Database Collation: utf8_general_ci
***** 2. row *****
      Db: my_db_name
      Name: Every_10_Minutes_Cleanup
```

```
Definer: root@localhost
Time zone: SYSTEM
Type: RECURRING
Execute at: NULL
Interval value: 10
Interval field: MINUTE
Starts: 2015-09-01 00:00:00
Ends: NULL
Status: ENABLED
Originator: 1
character_set_client: utf8
collation_connection: utf8_general_ci
Database Collation: utf8_general_ci
2 rows in set (0.06 sec)
```

## Des choses aléatoires à considérer

`DROP EVENT someEventName;` - Supprime l'événement et son code

`ON COMPLETION PRESERVE` - Lorsque l'événement est terminé, conservez-le. Sinon, il est supprimé.

Les événements sont comme des déclencheurs. Ils ne sont pas appelés par le programme d'un utilisateur. Ils sont plutôt programmés. En tant que tels, ils réussissent ou échouent silencieusement.

Le lien vers la page de manuel montre un certain degré de flexibilité avec les choix d'intervalles, présentés ci-dessous:

intervalle:

```
quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```

Les événements sont des mécanismes puissants qui gèrent les tâches récurrentes et planifiées pour votre système. Ils peuvent contenir autant d'énoncés, de routines DDL et DML, et de jointures compliquées que vous pouvez raisonnablement souhaiter. Veuillez consulter la page du manuel MySQL intitulée [Restrictions sur les programmes stockés](#) .

Lire Événements en ligne: <https://riptutorial.com/fr/mysql/topic/4319/evenements>

# Chapitre 21: Expressions régulières

## Introduction

Une expression régulière est un moyen puissant de spécifier un motif pour une recherche complexe.

## Exemples

### REGEXP / RLIKE

L' `REGEXP` (ou son synonyme, `RLIKE` ) permet la correspondance de modèle basée sur des expressions régulières.

Considérons la table d' `employee` suivante:

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | PHONE_NUMBER | SALARY   |
|-------------|------------|-----------|--------------|----------|
| 100         | Steven     | King      | 515.123.4567 | 24000.00 |
| 101         | Neena      | Kochhar   | 515.123.4568 | 17000.00 |
| 102         | Lex        | De Haan   | 515.123.4569 | 17000.00 |
| 103         | Alexander  | Hunold    | 590.423.4567 | 9000.00  |
| 104         | Bruce      | Ernst     | 590.423.4568 | 6000.00  |
| 105         | David      | Austin    | 590.423.4569 | 4800.00  |
| 106         | Valli      | Pataballa | 590.423.4560 | 4800.00  |
| 107         | Diana      | Lorentz   | 590.423.5567 | 4200.00  |
| 108         | Nancy      | Greenberg | 515.124.4569 | 12000.00 |
| 109         | Daniel     | Faviet    | 515.124.4169 | 9000.00  |
| 110         | John       | Chen      | 515.124.4269 | 8200.00  |

## Motif `^`

Sélectionnez tous les employés dont `FIRST_NAME` commence par **N**.

### Question

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP '^N'  
-- Pattern start with-----^
```

## Pattern `$ **`

Sélectionnez tous les employés dont `PHONE_NUMBER` se termine par **4569** .

### Question

```
SELECT * FROM employees WHERE PHONE_NUMBER REGEXP '4569$'  
-- Pattern end with-----^
```

## PAS REGEXP

Sélectionnez tous les employés dont `FIRST_NAME` *ne* commence *pas* par **N**.

### Question

```
SELECT * FROM employees WHERE FIRST_NAME NOT REGEXP '^N'  
-- Pattern does not start with-----^
```

## Regex Contient

Sélectionnez tous les employés dont `LAST_NAME` contient et dont `FIRST_NAME` contient `a`.

### Question

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP 'a' AND LAST_NAME REGEXP 'in'  
-- No ^ or $, pattern can be anywhere -----^
```

## Tout personnage entre []

Sélectionnez tous les employés dont `FIRST_NAME` commence par **A** ou **B** ou **C**.

### Question

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP '^[ABC]'  
-----^-----^-----^
```

## Motif ou |

Sélectionnez tous les employés dont `FIRST_NAME` commence par **A** ou **B** ou **C** et se termine par **r**, **e** ou **i**.

### Question

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP '^[ABC]|[rei]$'  
-----^-----^-----^
```

---

## Compter les expressions régulières

Considérez la requête suivante:

```
SELECT FIRST_NAME, FIRST_NAME REGEXP '^N' as matching FROM employees
```

`FIRST_NAME REGEXP '^N'` est 1 ou 0 selon que `FIRST_NAME` correspond à `^N`

Pour mieux le visualiser:

```
SELECT
FIRST_NAME,
IF(FIRST_NAME REGEXP '^N', 'matches ^N', 'does not match ^N') as matching
FROM employees
```

Enfin, comptez le nombre total de lignes correspondantes et non correspondantes avec:

```
SELECT
IF(FIRST_NAME REGEXP '^N', 'matches ^N', 'does not match ^N') as matching,
COUNT(*)
FROM employees
GROUP BY matching
```

Lire Expressions régulières en ligne: <https://riptutorial.com/fr/mysql/topic/9444/expressions-regulieres>



# Chapitre 22: Extraire les valeurs du type JSON

## Introduction

MySQL 5.7.8+ prend en charge le type JSON natif. Bien que vous puissiez créer des objets json de différentes manières, vous pouvez également accéder aux membres et les lire de différentes manières.

La fonction principale est `JSON_EXTRACT`, donc `->` et `->>` opérateurs sont plus conviviaux.

## Syntaxe

- `JSON_EXTRACT (json_doc, chemin d'accès [, ...])`
- `JSON_EXTRACT (json_doc, chemin)`
- `JSON_EXTRACT (json_doc, path1, path2)`

## Paramètres

| Paramètre             | La description       |
|-----------------------|----------------------|
| <code>json_doc</code> | document JSON valide |
| <code>chemin</code>   | chemin des membres   |

## Remarques

Mentionné dans [MySQL 5.7 Reference Manual](#)

- Plusieurs valeurs correspondantes par argument (s) de chemin

S'il est possible que ces arguments renvoient plusieurs valeurs, les valeurs correspondantes sont automatiquement récupérées sous la forme d'un tableau, dans l'ordre correspondant aux chemins qui les ont générés. Sinon, la valeur de retour est la seule valeur correspondante.

- `NULL` Résultat lorsque:
  - n'importe quelle dispute est `NULL`
  - chemin non assorti

Retourne `NULL` si un argument est `NULL` ou si aucun chemin ne recherche une valeur dans le document.

# Exemples

## Lire la valeur du tableau JSON

Créez la variable @myjson en tant que type JSON (en [savoir plus](#)):

```
SET @myjson = CAST('["A","B",{ "id":1,"label":"C"}]' as JSON) ;
```

SELECT des membres!

```
SELECT
  JSON_EXTRACT( @myjson , '$[1]' ) ,
  JSON_EXTRACT( @myjson , '$[*].label' ) ,
  JSON_EXTRACT( @myjson , '$[1].*' ) ,
  JSON_EXTRACT( @myjson , '$[2].*' )
;
-- result values:
'\\"B\\"', '\\\\"C\\"', NULL, '[1, \\"C\\"]'
-- visually:
"B", ["C"], NULL, [1, "C"]
```

## Opérateurs d'extraction JSON

Extraire le path par -> ou ->> Opérateurs, alors que ->> est la valeur UNQUOTED:

```
SELECT
  myjson_col->'${[1]}' , myjson_col->'${[1]}' ,
  myjson_col->>'${[*].label}' ,
  myjson_col->>'${[1].*}' ,
  myjson_col->>'${[2].*}'
FROM tablename ;
-- visuall:
  B, "B" , ["C"], NULL, [1, "C"]
__^^^ ^^
```

So col->>path est égal à JSON\_UNQUOTE(JSON\_EXTRACT(col,path)) :

Comme avec ->, l'opérateur ->> est toujours développé dans la sortie de EXPLAIN, comme le montre l'exemple suivant:

```
mysql> EXPLAIN SELECT c->>'$.name' AS name
->      FROM jemp WHERE g > 2\G
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: jemp
  partitions: NULL
         type: range
possible_keys: i
          key: i
         key_len: 5
          ref: NULL
         rows: 2
```

```
filtered: 100.00
  Extra: Using where
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
***** 1. row *****
Level: Note
Code: 1003
Message: /* select#1 */ select
json_unquote(json_extract(`jtest`.`jemp`.`c`, '$.name')) AS `name` from
`jtest`.`jemp` where (`jtest`.`jemp`.`g` > 2)
1 row in set (0.00 sec)
```

Lisez à propos de l' [extrait de chemin en ligne \(+\)](#)

Lire Extraire les valeurs du type JSON en ligne: <https://riptutorial.com/fr/mysql/topic/9042/extraire-les-valeurs-du-type-json>

# Chapitre 23: Fichiers journaux

## Exemples

### Une liste

- Journal général - toutes les requêtes - voir VARIABLE `general_log`
- Journal lent - requêtes plus longues que `long_query_time` - `slow_query_log_file`
- Binlog - pour la réplication et la sauvegarde - `log_bin_basename`
- Journal de relais - également pour la réplication
- erreurs générales - `mysqld.err`
- start / stop - `mysql.log` (pas très intéressant) - `log_error`
- InnoDB redo log - `iblog *`

Voir les variables `basedir` et `datadir` pour l'emplacement par défaut de nombreux journaux

Certains journaux sont activés / désactivés par d'autres variables. Certains sont soit écrits dans un fichier, soit dans une table.

(Note aux réviseurs: cela nécessite plus de détails et plus d'explications.)

*Documenteurs* : veuillez inclure l'emplacement et le nom par défaut pour chaque type de journal, à la fois pour Windows et \* nix. (Ou au moins autant que possible.)

### Journal de requête lent

Le journal de requêtes lent se compose d'événements de journal pour les requêtes prenant jusqu'à `long_query_time` secondes pour terminer. Par exemple, jusqu'à 10 secondes pour terminer. Pour voir le seuil de temps actuellement défini, émettez le message suivant:

```
SELECT @@long_query_time;
+-----+
| @@long_query_time |
+-----+
|          10.000000 |
+-----+
```

Il peut être défini en tant que variable GLOBAL, dans le fichier `my.cnf` ou `my.ini` . Ou il peut être défini par la connexion, bien que cela soit inhabituel. La valeur peut être réglée entre 0 et 10 (secondes). Quelle valeur utiliser?

- 10 est si haut qu'il est presque inutile;
- 2 est un compromis;
- 0,5 et d'autres fractions sont possibles;
- 0 capture tout; Cela pourrait remplir le disque dangereusement vite, mais peut être très utile.

La capture des requêtes lentes est activée ou désactivée. Et le fichier connecté est également

spécifié. Le ci-dessous capture ces concepts:

```
SELECT @@slow_query_log; -- Is capture currently active? (1=On, 0=Off)
SELECT @@slow_query_log_file; -- filename for capture. Resides in datadir
SELECT @@datadir; -- to see current value of the location for capture file

SET GLOBAL slow_query_log=0; -- Turn Off
-- make a backup of the Slow Query Log capture file. Then delete it.
SET GLOBAL slow_query_log=1; -- Turn it back On (new empty file is created)
```

Pour plus d'informations, s'il vous plaît voir la page de manuel MySQL [Le log des requêtes lentes](#)

Remarque: Les informations ci-dessus sur l'activation / désactivation du slowlog ont été modifiées dans 5.6 (?); l'ancienne version avait un autre mécanisme.

La "meilleure" façon de voir ce qui ralentit votre système:

```
long_query_time=...
turn on the slowlog
run for a few hours
turn off the slowlog (or raise the cutoff)
run pt-query-digest to find the 'worst' couple of queries. Or mysqldumpslow -s t
```

## Journal de requête général

Le journal de requête général contient une liste d'informations générales provenant des connexions client, des déconnexions et des requêtes. Il est inestimable pour le débogage, mais il constitue un obstacle à la performance (citation?).

Un exemple de vue d'un journal de requête général est présenté ci-dessous:

```
36 Query insert questions_c23(qId,ownerId,title,votes,answers,isClosed,closeVotes,views,owne
comments,answeredAccepted,askDate,closeDate,lastScanDate,ign,bn,pvtc,
mainTagForImport,prepStatus,touches,status,status_bef_change,cv_bef_change,max_cv_r
values(386666373, 1322183, 'How to post a numeric value in c#', 0, 1, 0, 0, 50, 1,
0, 0, '2016-07-29 19:40:32', null, now(), 0, 0, 0,
'c%23',0,1,'0','','0,0)
on duplicate key update title='How to post a numeric value in c#', votes=0, answers
answeredAccepted=0,lastScanDate=now(), touches=touches+1,status='0'
```

Pour déterminer si le journal général est en cours de capture:

```
SELECT @@general_log; -- 1 = Capture is active; 0 = It is not.
```

Pour déterminer le nom du fichier de capture:

```
SELECT @@general_log_file; -- Full path to capture file
```

Si le chemin d'accès complet au fichier n'est pas affiché, le fichier existe dans le `datadir`.

Exemple Windows:

```
+-----+
| @@general_log_file |
+-----+
| C:\ProgramData\MySQL\MySQL Server 5.7\Data\GuySmiley.log |
+-----+
```

Linux:

```
+-----+
| @@general_log_file |
+-----+
| /var/lib/mysql/ip-ww-xx-yy-zz.log |
+-----+
```

Lorsque des modifications sont apportées à la variable GLOBAL `general_log_file` , le nouveau journal est enregistré dans le `datadir` . Cependant, le chemin complet ne peut plus être reflété en examinant la variable.

Dans le cas où aucune entrée pour `general_log_file` ne `general_log_file` dans le fichier de configuration, la valeur par défaut est `@hostname.log` dans le `datadir` .

Les meilleures pratiques consistent à désactiver la capture. Enregistrez le fichier journal dans un répertoire de sauvegarde avec un nom de fichier reflétant l'heure de début / fin de la capture. La suppression du fichier précédent si un *déplacement du* système de fichiers n'a pas eu lieu. Établissez un nouveau nom de fichier pour le fichier journal et activez la capture (tous affichés ci-dessous). Les meilleures pratiques incluent également une détermination minutieuse si vous souhaitez même capturer pour le moment. En règle générale, la capture est activée à des fins de débogage uniquement.

Un nom de fichier typique d'un système de fichiers pour un journal sauvegardé peut être:

```
/LogBackup/GeneralLog_20160802_1520_to_20160802_1815.log
```

où la date et l'heure font partie du nom de fichier sous forme de plage.

Pour Windows, notez la séquence suivante avec les modifications de réglage.

```
SELECT @@general_log; -- 0. Not being captured
SELECT @@general_log_file; -- C:\ProgramData\MySQL\MySQL Server 5.6\Data\GuySmiley.log
SELECT @@datadir; -- C:\ProgramData\MySQL\MySQL Server 5.7\Data\
SET GLOBAL general_log_file='GeneralLogBegin_20160803_1420.log'; -- datetime clue
SET GLOBAL general_log=1; -- Turns on actual log capture. File is created under `datadir`
SET GLOBAL general_log=0; -- Turn logging off
```

Linux est similaire. Celles-ci représenteraient des changements dynamiques. Tout redémarrage du serveur prendrait les paramètres du fichier de configuration.

En ce qui concerne le fichier de configuration, tenez compte des paramètres de variable pertinents suivants:

```
[mysqld]
```

```
general_log_file = /path/to/currentquery.log
general_log      = 1
```

De plus, la variable `log_output` peut être configurée pour la sortie `TABLE`, pas seulement `FILE`. Pour cela, veuillez consulter [Destinations](#).

Veuillez consulter la page du manuel MySQL [Le journal des requêtes générales](#).

## Journal des erreurs

Le journal des erreurs contient les informations de démarrage et d'arrêt et les événements critiques rencontrés par le serveur.

Voici un exemple de son contenu:

```
2016-08-02 20:40:39 2420 [Note] Shutting down plugin 'binlog'
2016-08-02 20:40:39 2420 [Note] mysqld: Shutdown complete

2016-08-02 20:43:11 2888 [Note] Plugin 'FEDERATED' is disabled.
2016-08-02 20:43:11 2888 [Note] InnoDB: Using atomics to ref count buffer pool pages
2016-08-02 20:43:11 2888 [Note] InnoDB: The InnoDB memory heap is disabled
```

La variable `log_error` contient le chemin d'accès au fichier journal pour la journalisation des erreurs.

En l'absence d'une entrée de fichier de configuration pour `log_error`, le système `log_error` ses valeurs par défaut pour `@@hostname.err` dans le `datadir`. Notez que `log_error` n'est pas une variable dynamique. En tant que tel, les modifications sont effectuées via des modifications de fichier CNF ou INI et un redémarrage du serveur (ou en affichant "Rinçage et modification du nom du fichier journal des erreurs" dans le lien Page de manuel en bas).

La journalisation ne peut pas être désactivée pour des erreurs. Ils sont importants pour la santé du système lors de la résolution des problèmes. En outre, les entrées sont peu fréquentes par rapport au journal de requête général.

La variable GLOBAL `log_warnings` définit le niveau de verbosité qui varie selon la version du serveur. L'extrait suivant illustre:

```
SELECT @@log_warnings; -- make a note of your prior setting
SET GLOBAL log_warnings=2; -- setting above 1 increases output (see server version)
```

`log_warnings` comme vu ci-dessus est une variable dynamique.

Les modifications apportées au fichier de configuration dans les fichiers `cnf` et `ini` peuvent ressembler à ceci.

```
[mysqld]
log_error      = /path/to/CurrentError.log
log_warnings   = 2
```

MySQL 5.7.2 a étendu la verbosité du niveau d'avertissement à 3 et a ajouté la valeur GLOBAL `log_error_verbosity` . Encore une fois, il a été [introduit](#) dans 5.7.2. Il peut être défini dynamiquement et vérifié en tant que variable ou défini via les paramètres du fichier de configuration `cnf` ou `ini` .

À partir de MySQL 5.7.2:

```
[mysqld]
log_error          = /path/to/CurrentError.log
log_warnings      = 2
log_error_verbosity = 3
```

S'il vous plaît voir le manuel MySQL page intitulée [Le journal d'erreur](#) en particulier pour Flushing et en renommant le fichier journal d'erreur, et son *erreur* section *Journal verbosité* avec les versions liées à `log_warnings` et `error_log_verbosity` .

**Lire Fichiers journaux en ligne:** <https://riptutorial.com/fr/mysql/topic/5102/fichiers-journaux>



---

# Chapitre 24: Groupage

## Exemples

### La désambiguïsation

"MySQL Cluster" désambiguïsation ...

- NDB Cluster - Un moteur spécialisé, principalement en mémoire. Pas largement utilisé.
- Cluster Galera alias Percona XtraDB Cluster aka PXC alias MariaDB avec Galera. - une très bonne solution de haute disponibilité pour MySQL; cela va au-delà de la réplication.

Voir les pages individuelles sur ces variantes de "Cluster".

Pour "index clusterisé", voir page (s) sur `PRIMARY KEY` .

Lire Groupage en ligne: <https://riptutorial.com/fr/mysql/topic/5130/groupage>

---

# Chapitre 25: Index et clés

## Syntaxe

- - Créer un index simple

```
CREATE INDEX nom_index ON nom_table ( nom_colonne1 [, nom_colonne2 , ...])
```

- - Créer un index unique

```
CREATE UNIQUE INDEX nom_index ON nom_table ( nom_colonne1 [, nom_colonne2 , ...])
```

- - Index de baisse

```
DROP INDEX nom_index ON nom_table [algorithm_option | lock_option ] ...
```

```
option_algorithme : ALGORITHM [=] {DEFAULT | INPLACE | COPY}
```

```
lock_option: LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
```

## Remarques

---

## Les concepts

Un index dans une table MySQL fonctionne comme un index dans un livre.

Disons que vous avez un livre sur les bases de données et que vous souhaitez trouver des informations sur, disons, le stockage. Sans index (sans autre aide, comme une table des matières), vous devez parcourir les pages une à une jusqu'à ce que vous trouviez le sujet (c'est-à-dire une "analyse complète de la table"). D'un autre côté, un index a une liste de mots-clés, vous devriez donc consulter l'index et voir que le stockage est mentionné aux pages 113-120, 231 et 354. Ensuite, vous pouvez retourner directement à ces pages, sans chercher (c'est une recherche avec un index, un peu plus rapide).

Bien sûr, l'utilité de l'index dépend de plusieurs facteurs - quelques exemples, en utilisant la comparaison ci-dessus:

- Si vous aviez un livre sur les bases de données et indexé le mot "base de données", vous pourriez voir qu'il est mentionné aux pages 1-59, 61-290 et 292-400. Cela fait beaucoup de pages, et dans un tel cas, l'index n'est pas d'une grande aide et il peut être plus rapide de parcourir les pages une par une. (Dans une base de données, il s'agit d'une "mauvaise sélectivité".)
- Pour un livre de 10 pages, cela n'a aucun sens de créer un index, car vous pourriez vous retrouver avec un livre de 10 pages préfixé par un index de 5 pages, ce qui est stupide - il suffit de scanner les 10 pages et de les terminer .

- L'index doit également être utile - il est généralement inutile d'indexer, par exemple, la fréquence de la lettre "L" par page.

## Exemples

### Créer un index

```
-- Create an index for column 'name' in table 'my_table'  
CREATE INDEX idx_name ON my_table(name);
```

### Créer un index unique

Un index unique empêche l'insertion de données dupliquées dans une table. `NULL` valeurs `NULL` peuvent être insérées dans les colonnes qui font partie de l'index unique (car, par définition, une valeur `NULL` est différente de toute autre valeur, y compris une autre valeur `NULL` )

```
-- Creates a unique index for column 'name' in table 'my_table'  
CREATE UNIQUE INDEX idx_name ON my_table(name);
```

### Index de baisse

```
-- Drop an index for column 'name' in table 'my_table'  
DROP INDEX idx_name ON my_table;
```

### Créer un index composite

Cela créera un index composite des deux clés, `mystring` et `mydatetime` et accélérera les requêtes avec les deux colonnes de la clause `WHERE` .

```
CREATE INDEX idx_mycol_myothercol ON my_table(mycol, myothercol)
```

**Note:** la commande est importante! Si la requête de recherche n'inclut pas les deux colonnes dans la clause `WHERE` , elle ne peut utiliser que l'index le plus à gauche. Dans ce cas, une requête avec `mycol` dans le `WHERE` utilisera l'index, une requête recherchant `myothercol` **sans** rechercher également `mycol` **ne le sera pas** . Pour plus d'informations, [consultez ce billet de blog](#) .

**Note:** En raison de la façon dont fonctionne BTREE, les colonnes généralement interrogées dans les plages doivent aller dans la valeur la plus à droite. Par exemple, les colonnes `DATETIME` sont généralement interrogées comme `WHERE datecol > '2016-01-01 00:00:00'` . Les index BTREE gèrent les plages de manière très efficace, mais uniquement si la colonne interrogée en tant qu'intervalle est la dernière de l'index composite.

### Touche AUTO\_INCREMENT

```
CREATE TABLE (  
  id INT UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
...  
PRIMARY KEY(id),  
... );
```

### Notes principales:

- Commence par 1 et s'incrémente automatiquement de 1 lorsque vous ne parvenez pas à le spécifier sur `INSERT` ou spécifiez-le comme `NULL` .
- Les identifiants sont toujours distincts les uns des autres, mais ...
- Ne faites pas d'hypothèses (pas de lacunes, générées consécutivement, non réutilisées, etc.) sur les valeurs de l'ID, sauf qu'elles sont uniques à un instant donné.

### Notes subtiles:

- Au redémarrage du serveur, la valeur «next» est «calculée» en tant que `MAX(id)+1` .
- Si la dernière opération avant l'arrêt ou la panne était de supprimer l'ID le plus élevé, cet identifiant *peut* être réutilisé (cela dépend du moteur). Donc, *ne faites pas confiance à `auto_increments` pour être unique en permanence* ; ils sont uniques à tout moment.
- Pour les solutions multi-maîtres ou en cluster, voir `auto_increment_offset` et `auto_increment_increment` .
- C'est bien d'avoir autre chose que la `PRIMARY KEY` et simplement faire `INDEX(id)` . (Ceci est une optimisation dans certaines situations.)
- L'utilisation de `AUTO_INCREMENT` comme "clé de `PARTITION` " est rarement bénéfique. faire quelque chose de différent.
- Diverses opérations *peuvent* "brûler" des valeurs. Cela se produit quand ils pré-allouent des valeurs, alors ne les utilisez pas: `INSERT IGNORE` (avec la clé dup), `REPLACE` (qui est `DELETE` plus `INSERT` ) et d'autres. `ROLLBACK` est une autre cause des lacunes dans les identifiants.
- Dans Replication, vous ne pouvez pas faire confiance aux identifiants pour arriver au (x) esclave (s) dans l'ordre croissant. Bien que les identifiants soient attribués dans un ordre consécutif, les instructions InnoDB sont envoyées aux esclaves dans l'ordre `COMMIT` .

Lire Index et clés en ligne: <https://riptutorial.com/fr/mysql/topic/1748/index-et-cles>

# Chapitre 26: Informations sur le serveur

## Paramètres

| Paramètres | Explication   |
|------------|---|
| GLOBAL     | Affiche les variables telles qu'elles sont configurées pour le serveur entier. Optionnel. |
| SESSION    | Affiche les variables configurées pour cette session uniquement. Optionnel.               |

## Exemples

### SHOW VARIABLES exemple

Pour obtenir toutes les variables du serveur, exécutez cette requête dans la fenêtre SQL de votre interface préférée (PHPMysqlAdmin ou autre) ou dans l'interface CLI MySQL.

```
SHOW VARIABLES;
```

Vous pouvez spécifier si vous voulez les variables de session ou les variables globales comme suit:

Variables de session:

```
SHOW SESSION VARIABLES;
```

Variables globales:

```
SHOW GLOBAL VARIABLES;
```

Comme toute autre commande SQL, vous pouvez ajouter des paramètres à votre requête, tels que la commande LIKE:

```
SHOW [GLOBAL | SESSION] VARIABLES LIKE 'max_join_size';
```

Ou, en utilisant des caractères génériques:

```
SHOW [GLOBAL | SESSION] VARIABLES LIKE '%size%';
```

Vous pouvez également filtrer les résultats de la requête SHOW en utilisant un paramètre WHERE comme suit:

```
SHOW [GLOBAL | SESSION] VARIABLES WHERE VALUE > 0;
```

## Exemple de SHOW STATUS

Pour obtenir l'état du serveur de base de données, exécutez cette requête dans la fenêtre SQL de votre interface préférée (PHPMyAdmin ou autre) ou sur l'interface CLI MySQL.

```
SHOW STATUS;
```

Vous pouvez spécifier si vous souhaitez recevoir le statut SESSION ou GLOBAL de votre serveur comme suit: Statut de la session:

```
SHOW SESSION STATUS;
```

Statut global:

```
SHOW GLOBAL STATUS;
```

Comme toute autre commande SQL, vous pouvez ajouter des paramètres à votre requête, tels que la commande LIKE:

```
SHOW [GLOBAL | SESSION] STATUS LIKE 'Key%';
```

Ou la commande Where:

```
SHOW [GLOBAL | SESSION] STATUS WHERE VALUE > 0;
```

La principale différence entre GLOBAL et SESSION est que, avec le modificateur GLOBAL, la commande affiche des informations agrégées sur le serveur et toutes ses connexions, tandis que le modificateur SESSION affiche uniquement les valeurs de la connexion en cours.

Lire Informations sur le serveur en ligne: <https://riptutorial.com/fr/mysql/topic/9924/informations-sur-le-serveur>

# Chapitre 27: INSÉRER

## Syntaxe

1. `INSERT [LOW_PRIORITY | RETARDÉ | HIGH_PRIORITY] [IGNORE] [INTO] nom_de_table [PARTITION (nom_partition, ...)] [(nom_colonne, ...)] {VALUES | VALUE} ({expr | DEFAULT}, ...), (...), ... [ON DUPLICATE KEY UPDATE nom_colonne = expr [, nom_colonne = expr] ...]`
2. `INSERT [LOW_PRIORITY | RETARDÉ | HIGH_PRIORITY] [IGNORE] [INTO] nom_de_table [PARTITION (nom_partition, ...)] SET nom_col = {expr | DEFAULT}, ... [ON DUPLICATE KEY UPDATE nom_colonne = expr [, nom_colonne = expr] ...]`
3. `INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE] [INTO] tbl_name [PARTITION (nom_partition, ...)] [(nom_colonne, ...)] SELECT ... [ON DUPLICATE KEY UPDATE nom_colonne = expr [, nom_colonne = expr] ...]`
4. Une expression `expr` peut faire référence à n'importe quelle colonne définie précédemment dans une liste de valeurs. Par exemple, vous pouvez le faire car la valeur de `col2` fait référence à `col1`, qui a déjà été affecté:  
`INSERT INTO tbl_name (col1, col2) VALUES (15, col1 * 2);`
5. Les instructions `INSERT` qui utilisent la syntaxe `VALUES` peuvent insérer plusieurs lignes. Pour ce faire, incluez plusieurs listes de valeurs de colonne, chacune entre parenthèses et séparées par des virgules. Exemple:  
`INSERT INTO tbl_name (a, b, c) VALUES (1,2,3), (4,5,6), (7,8,9);`
6. La liste de valeurs pour chaque ligne doit figurer entre parenthèses. L'instruction suivante est illégale car le nombre de valeurs dans la liste ne correspond pas au nombre de noms de colonnes:  
`INSERT INTO nom_de_table (a, b, c) VALUES (1,2,3,4,5,6,7,8,9);`
7. **INSERT ... SELECT Syntaxe**  
`INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE] [INTO] tbl_name [PARTITION (nom_partition, ...)] [(nom_colonne, ...)] SELECT ... [ON DUPLICATE KEY UPDATE nom_colonne = expr, ...]`
8. Avec `INSERT ... SELECT`, vous pouvez rapidement insérer plusieurs lignes dans une table à partir d'une ou plusieurs tables. Par exemple:  
`INSERT INTO tbl_temp2 (fld_id) SELECT tbl_temp1.fld_order_id FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;`

## Remarques

[Syntaxe INSERT officielle](#)

# Exemples

## Insert de base

```
INSERT INTO `table_name` (`field_one`, `field_two`) VALUES ('value_one', 'value_two');
```

Dans cet exemple trivial, `table_name` est l'endroit où les données doivent être ajoutées, `field_one` et `field_two` sont des champs pour définir les données, et `value_one` et `value_two` sont les données à faire respectivement avec `field_one` et `field_two`.

Il est recommandé de répertorier les champs dans lesquels vous insérez des données dans votre code, comme si la table était modifiée et que de nouvelles colonnes étaient ajoutées, votre insertion serait rompue si elles ne s'y trouvaient pas.

## INSERT, ON DUPLICATE MISE À JOUR DE LA CLÉ

```
INSERT INTO `table_name`
  (`index_field`, `other_field_1`, `other_field_2`)
VALUES
  ('index_value', 'insert_value', 'other_value')
ON DUPLICATE KEY UPDATE
  `other_field_1` = 'update_value',
  `other_field_2` = VALUES(`other_field_2`);
```

Cela `INSERT` dans `table_name` les valeurs spécifiées, mais si la clé unique existe déjà, il mettra à jour le `other_field_1` d'avoir une nouvelle valeur.

Parfois, lors de la mise à jour sur une clé dupliquée, il est utile d'utiliser `VALUES()` pour accéder à la valeur d'origine transmise à `INSERT` au lieu de définir directement la valeur. De cette façon, vous pouvez définir différentes valeurs en utilisant `INSERT` et `UPDATE`. Voir l'exemple ci-dessus où `other_field_1` est défini sur `insert_value` sur `INSERT` ou `update_value` sur `UPDATE` alors que `other_field_2` est toujours défini sur `other_value`.

Pour que l'insertion sur la mise à jour des clés en double (IODKU) fonctionne, il faut que le schéma contienne une clé unique qui signalera un conflit en double. Cette clé unique peut être une clé primaire ou non. Il peut s'agir d'une clé unique sur une seule colonne ou d'une multi-colonne (clé composite).

## Insertion de plusieurs lignes

```
INSERT INTO `my_table` (`field_1`, `field_2`) VALUES
  ('data_1', 'data_2'),
  ('data_1', 'data_3'),
  ('data_4', 'data_5');
```

C'est un moyen facile d'ajouter plusieurs lignes à la fois avec une seule instruction `INSERT`.

Ce type d'insertion par lots est beaucoup plus rapide que l'insertion de lignes une par une. En règle générale, l'insertion de 100 lignes dans une seule insertion de lot est 10 fois plus rapide que



si vous les insérez toutes individuellement.

## Ignorer les lignes existantes

Lors de l'importation de jeux de données volumineux, il peut être préférable, dans certaines circonstances, d'ignorer les lignes entraînant l'échec de la requête en raison d'une restriction de colonne, par exemple, les clés primaires dupliquées. Cela peut être fait en utilisant `INSERT IGNORE`.

Prenons l'exemple suivant:

```
SELECT * FROM `people`;  
--- Produces:  
+----+-----+  
| id | name |  
+----+-----+  
| 1  | john |  
| 2  | anna |  
+----+-----+  
  
INSERT IGNORE INTO `people` (`id`, `name`) VALUES  
    ('2', 'anna'), --- Without the IGNORE keyword, this record would produce an error  
    ('3', 'mike');
```

```
SELECT * FROM `people`;  
--- Produces:  
+----+-----+  
| id | name |  
+----+-----+  
| 1  | john |  
| 2  | anna |  
| 3  | mike |  
+----+-----+
```

La chose importante à retenir est que *INSERT IGNORE* ignorera également les autres erreurs en silence. Voici ce que disent les documentations officielles de MySQL:

Les conversions de données qui déclencheraient des erreurs annulent l'instruction si `IGNORE` n'est pas spécifié. Avec `IGNORE`, les valeurs non valides sont ajustées aux valeurs les plus proches et insérées; des avertissements sont produits mais l'instruction ne s'interrompt pas.

**Remarque: - La section ci-dessous est ajoutée pour des raisons d'exhaustivité, mais n'est pas considérée comme la meilleure pratique (cela échouerait, par exemple, si une autre colonne était ajoutée dans la table).**

Si vous spécifiez la valeur de la colonne correspondante pour toutes les colonnes de la table, vous pouvez ignorer la liste de colonnes dans l'instruction `INSERT` comme suit:

```
INSERT INTO `my_table` VALUES  
    ('data_1', 'data_2'),  
    ('data_1', 'data_3'),  
    ('data_4', 'data_5');
```

## INSERT SELECT (Insérer des données d'une autre table)

C'est la méthode de base pour insérer des données d'une autre table avec l'instruction SELECT.

```
INSERT INTO `tableA` (`field_one`, `field_two`)
  SELECT `tableB`.`field_one`, `tableB`.`field_two`
  FROM `tableB`
  WHERE `tableB`.clmn <> 'someValue'
  ORDER BY `tableB`.`sorting_clmn`;
```

Vous pouvez `SELECT * FROM`, mais `tableA` et `tableB` devez avoir correspondre le nombre de colonnes et types de données correspondant.

Les colonnes avec `AUTO_INCREMENT` sont traitées comme dans la clause `INSERT with VALUES`.

Cette syntaxe facilite le remplissage de tables (temporaires) avec des données provenant d'autres tables, d'autant plus lorsque les données doivent être filtrées sur l'insert.

## INSERT avec AUTO\_INCREMENT + LAST\_INSERT\_ID ()

Lorsqu'une table a une `PRIMARY KEY AUTO_INCREMENT`, normalement, on ne l'insère pas dans cette colonne. Au lieu de cela, spécifiez toutes les autres colonnes, puis demandez quel était le nouvel identifiant.

```
CREATE TABLE t (
  id SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL,
  this ...,
  that ...,
  PRIMARY KEY(id) );

INSERT INTO t (this, that) VALUES (... , ...);
SELECT LAST_INSERT_ID() INTO @id;
INSERT INTO another_table (... , t_id, ...) VALUES (... , @id, ...);
```

Notez que `LAST_INSERT_ID()` est lié à la session, donc même si plusieurs connexions sont insérées dans la même table, chacune avec son propre identifiant.

Votre API client a probablement une autre manière d'obtenir le `LAST_INSERT_ID()` sans effectuer réellement une commande `SELECT` et remettre la valeur au client au lieu de la laisser dans une `@variable` dans MySQL. C'est généralement préférable.

### Exemple plus long et plus détaillé

L'utilisation "normale" d'IODKU consiste à déclencher une "clé dupliquée" basée sur une clé `UNIQUE`, et non sur la touche `AUTO_INCREMENT PRIMARY KEY`. Ce qui suit illustre ceci. Notez qu'il ne fournit pas l' `id` dans l'INSERT.

Configuration des exemples à suivre:

```
CREATE TABLE iodku (
  id INT AUTO_INCREMENT NOT NULL,
```

```

name VARCHAR(99) NOT NULL,
misc INT NOT NULL,
PRIMARY KEY(id),
UNIQUE(name)
) ENGINE=InnoDB;

INSERT INTO iodku (name, misc)
VALUES
('Leslie', 123),
('Sally', 456);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
+----+-----+-----+
| id | name  | misc |
+----+-----+-----+
|  1 | Leslie | 123 |
|  2 | Sally  | 456 |
+----+-----+-----+

```

Le cas où IODKU effectue une "mise à jour" et `LAST_INSERT_ID()` récupère l' id correspondant:

```

INSERT INTO iodku (name, misc)
VALUES
('Sally', 3333)           -- should update
ON DUPLICATE KEY UPDATE  -- `name` will trigger "duplicate key"
id = LAST_INSERT_ID(id),
misc = VALUES(misc);
SELECT LAST_INSERT_ID();  -- picking up existing value
+-----+
| LAST_INSERT_ID() |
+-----+
|                2 |
+-----+

```

Le cas où IODKU effectue un "insert" et `LAST_INSERT_ID()` récupère le nouvel id :

```

INSERT INTO iodku (name, misc)
VALUES
('Dana', 789)           -- Should insert
ON DUPLICATE KEY UPDATE
id = LAST_INSERT_ID(id),
misc = VALUES(misc);
SELECT LAST_INSERT_ID();  -- picking up new value
+-----+
| LAST_INSERT_ID() |
+-----+
|                3 |
+-----+

```

Contenu de la table résultant:

```

SELECT * FROM iodku;
+----+-----+-----+
| id | name  | misc |
+----+-----+-----+
|  1 | Leslie | 123 |
|  2 | Sally  | 3333 | -- IODKU changed this

```

```
| 3 | Dana | 789 | -- IODKU added this
+----+-----+-----+
```

## Identifiants AUTO\_INCREMENT perdus

Plusieurs fonctions d'insertion peuvent "graver" des identifiants. Voici un exemple d'utilisation d'InnoDB (les autres moteurs peuvent fonctionner différemment):

```
CREATE TABLE Burn (
  id SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL,
  name VARCHAR(99) NOT NULL,
  PRIMARY KEY(id),
  UNIQUE(name)
) ENGINE=InnoDB;

INSERT IGNORE INTO Burn (name) VALUES ('first'), ('second');
SELECT LAST_INSERT_ID();           -- 1
SELECT * FROM Burn ORDER BY id;
+----+-----+
| 1 | first |
| 2 | second |
+----+-----+

INSERT IGNORE INTO Burn (name) VALUES ('second'); -- dup 'IGNOREd', but id=3 is burned
SELECT LAST_INSERT_ID();           -- Still "1" -- can't trust in this situation
SELECT * FROM Burn ORDER BY id;
+----+-----+
| 1 | first |
| 2 | second |
+----+-----+

INSERT IGNORE INTO Burn (name) VALUES ('third');
SELECT LAST_INSERT_ID();           -- now "4"
SELECT * FROM Burn ORDER BY id;    -- note that id=3 was skipped over
+----+-----+
| 1 | first |
| 2 | second |
| 4 | third | -- notice that id=3 has been 'burned'
+----+-----+
```

Pensez-y (grosso modo) de cette façon: tout d'abord, l'insert regarde le nombre de lignes *pouvant* être insérées. Ensuite, récupérez les nombreuses valeurs de auto\_increment pour cette table. Enfin, insérez les lignes, en utilisant les identifiants si nécessaire et en gravant tous les restes.

La seule fois où le reste est récupérable est si le système est arrêté et redémarré. Au redémarrage, `MAX(id)` est effectivement exécuté. Cela peut réutiliser les identifiants qui ont été gravés ou qui ont été libérés par `DELETES` des identifiants les plus élevés.

Essentiellement, toute version d' `INSERT` (y compris `REPLACE`, qui est `DELETE + INSERT`) peut graver des identifiants. Dans InnoDB, la variable globale (pas de session!) `innodb_autoinc_lock_mode` peut être utilisée pour contrôler une partie de ce qui se passe.

Lors de la "normalisation" de longues chaînes dans un `AUTO INCREMENT id`, la gravure peut facilement se produire. Cela *pourrait* entraîner un débordement de la taille de l' `INT` vous avez

choisi.

Lire INSÉRER en ligne: <https://riptutorial.com/fr/mysql/topic/866/insérer>

# Chapitre 28: Installez le conteneur Mysql avec Docker-Compose

## Exemples

### Exemple simple avec docker-compose

Ceci est un exemple simple pour créer un serveur mysql avec docker

1.- créer **docker-compose.yml** :

**Remarque:** Si vous souhaitez utiliser le même conteneur pour tous vos projets, vous devez créer un PATH dans votre HOME\_PATH. Si vous souhaitez le créer pour chaque projet, vous pouvez créer un répertoire **Docker** dans votre projet.

```
version: '2'
services:
  cabin_db:
    image: mysql:latest
    volumes:
      - "./.mysql-data/db:/var/lib/mysql"
    restart: always
    ports:
      - 3306:3306
    environment:
      MYSQL_ROOT_PASSWORD: rootpw
      MYSQL_DATABASE: cabin
      MYSQL_USER: cabin
      MYSQL_PASSWORD: cabinpw
```

2.- lancez-le:

```
cd PATH_TO_DOCKER-COMPOSE.YML
docker-compose up -d
```

3.- se connecter au serveur

```
mysql -h 127.0.0.1 -u root -P 3306 -p rootpw
```

Hourra!!

4.- arrêter le serveur

```
docker-compose stop
```

Lire Installez le conteneur Mysql avec Docker-Compose en ligne:

<https://riptutorial.com/fr/mysql/topic/4458/installez-le-conteneur-mysql-avec-docker-compose>

# Chapitre 29: Jeux de caractères et classements

## Exemples

### Déclaration

```
CREATE TABLE foo ( ...
  name CHARACTER SET utf8mb4
  ... );
```

### Connexion

Pour utiliser les jeux de caractères, il est essentiel de dire au serveur MySQL quel est le codage des octets du client. Voici un moyen:

```
SET NAMES utf8mb4;
```

Chaque langage (PHP, Python, Java, ...) a sa propre façon de le faire généralement préférable à `SET NAMES .`

Par exemple: `SET NAMES utf8mb4`, avec une colonne déclarée `CHARACTER SET latin1` - cela convertira de latin1 en utf8mb4 quand `INSERTing` et reconvertira en `SELECTing`.

### Quel jeu de caractères et quelle collection?

Il existe des dizaines de jeux de caractères avec des centaines de classements. (Un classement donné appartient à un seul jeu de caractères.) Voir la sortie de `SHOW COLLATION;`.

Il n'y a généralement que 4 `CHARACTER SETs` importants:

```
ascii -- basic 7-bit codes.
latin1 -- ascii, plus most characters needed for Western European languages.
utf8 -- the 1-, 2-, and 3-byte subset of utf8. This excludes Emoji and some of Chinese.
utf8mb4 -- the full set of UTF8 characters, covering all current languages.
```

Tous incluent des caractères anglais, codés de manière identique. `utf8` est un sous-ensemble de `utf8mb4`.

Meilleur entraînement...

- Utilisez `utf8mb4` pour toute colonne `TEXT` ou `VARCHAR` pouvant contenir plusieurs langues.
- Utilisez `ascii` (`latin1` est ok) pour les chaînes hexadécimales (UUID, MD5, etc.) et les codes simples (`country_code`, `postal_code`, etc.).

`utf8mb4` n'existait pas avant la version 5.5.3, donc `utf8` était le meilleur disponible avant cela.

En dehors de MySQL , "UTF8" signifie les mêmes choses que MySQL, mais pas MySQL.

Les classements commencent par le nom du jeu de caractères et se terminent généralement par `_ci` pour "case and accent `_bin` " ou `_bin` pour "comparez simplement les bits.

Le dernier classement `utf8mb4` est `utf8mb4_unicode_520_ci` , basé sur Unicode 5.20. Si vous travaillez avec une seule langue, vous pouvez, par exemple, `utf8mb4_polish_ci` , qui réorganisera légèrement les lettres en fonction des conventions polonaises.

## Définition des jeux de caractères sur les tables et les champs

Vous pouvez définir un jeu de caractères à la fois par table, ainsi que par champ individuel à l'aide des instructions `CHARACTER SET` et `CHARSET` :

```
CREATE TABLE Address (  
  `AddressID`    INTEGER NOT NULL PRIMARY KEY,  
  `Street`      VARCHAR(80) CHARACTER SET ASCII,  
  `City`        VARCHAR(80),  
  `Country`     VARCHAR(80) DEFAULT "United States",  
  `Active`      BOOLEAN DEFAULT 1,  
) Engine=InnoDB default charset=UTF8;
```

`City` et `Country` utiliseront `UTF8` , car nous le définissons comme jeu de caractères par défaut pour la table. `Street` , par contre, utilisera `ASCII` , comme nous lui avons expressément demandé de le faire.

Définir le bon jeu de caractères dépend fortement de votre jeu de données, mais peut également améliorer considérablement la portabilité entre les systèmes travaillant avec vos données.

Lire Jeux de caractères et classements en ligne: <https://riptutorial.com/fr/mysql/topic/4569/jeux-de-caracteres-et-classements>



---

# Chapitre 30: JOINS: Rejoignez 3 table avec le même nom d'id.

## Exemples

### Rejoignez 3 tables sur une colonne du même nom

```
CREATE TABLE Table1 (  
  id INT UNSIGNED NOT NULL,  
  created_on DATE NOT NULL,  
  PRIMARY KEY (id)  
)  
CREATE TABLE Table2 (  
  id INT UNSIGNED NOT NULL,  
  personName VARCHAR(255) NOT NULL,  
  PRIMARY KEY (id)  
)  
CREATE TABLE Table3 (  
  id INT UNSIGNED NOT NULL,  
  accountName VARCHAR(255) NOT NULL,  
  PRIMARY KEY (id)  
)
```

après avoir créé les tables, vous pouvez faire une requête select pour obtenir les identifiants des trois tables identiques

```
SELECT  
  t1.id AS table1Id,  
  t2.id AS table2Id,  
  t3.id AS table3Id  
FROM Table1 t1  
LEFT JOIN Table2 t2 ON t2.id = t1.id  
LEFT JOIN Table3 t3 ON t3.id = t1.id
```

Lire JOINS: Rejoignez 3 table avec le même nom d'id. en ligne:

<https://riptutorial.com/fr/mysql/topic/9921/joins--rejoignez-3-table-avec-le-meme-nom-d-id->

# Chapitre 31: Joint

## Syntaxe

- `INNER` et `OUTER` sont ignorés.
- `FULL` n'est pas implémenté dans MySQL.
- "commajoin" ( `FROM a,b WHERE ax=by` ) est mal vu; utilisez `FROM a JOIN b ON ax=by` place.
- `FROM a JOIN b ON ax = by` inclut les lignes qui correspondent dans les deux tables.
- `FROM a LEFT JOIN b ON ax = by` inclut toutes les lignes de `a` , plus les données correspondantes de `b` ou `NULLs` s'il n'y a pas de ligne correspondante.

## Exemples

### Exemples de jointure

Requête pour créer une table sur la base de données

```
CREATE TABLE `user` (  
  `id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,  
  `name` varchar(30) NOT NULL,  
  `course` smallint(5) unsigned DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB;  
  
CREATE TABLE `course` (  
  `id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,  
  `name` varchar(50) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB;
```

Puisque nous utilisons les tables InnoDB et que nous savons que `user.course` et `course.id` sont liés, nous pouvons spécifier une relation de clé étrangère:

```
ALTER TABLE `user`  
ADD CONSTRAINT `FK_course`  
FOREIGN KEY (`course`) REFERENCES `course` (`id`)  
ON UPDATE CASCADE;
```

### Rejoindre Query (Inner Join)

```
SELECT user.name, course.name  
FROM `user`  
INNER JOIN `course` on user.course = course.id;
```

### JOIN avec sous-requête (table "Derived")

```

SELECT x, ...
  FROM ( SELECT y, ... FROM ... ) AS a
 JOIN tbl ON tbl.x = a.y
 WHERE ...

```

Cela évaluera la sous-requête dans une table temporaire, puis `JOIN` ce à `tbl`.

Avant 5.6, il ne pouvait y avoir d'index sur la table temporaire. Donc, c'était potentiellement très inefficace:

```

SELECT ...
  FROM ( SELECT y, ... FROM ... ) AS a
 JOIN ( SELECT x, ... FROM ... ) AS b ON b.x = a.y
 WHERE ...

```

Avec 5.6, l'optimiseur calcule le meilleur index et le crée à la volée. (Cela a un peu de frais, alors ce n'est pas encore parfait.)

Un autre paradigme commun est d'avoir une sous-requête pour initialiser quelque chose:

```

SELECT
    @n := @n + 1,
    ...
  FROM ( SELECT @n := 0 ) AS initialize
 JOIN the_real_table
 ORDER BY ...

```

(Remarque: il s'agit techniquement d'un produit `CROSS JOIN` (produit cartésien), comme l'indique l'absence de la valeur `ON`. Toutefois, elle est efficace car la sous-requête ne renvoie qu'une ligne qui doit correspondre aux `n` lignes de `the_real_table`.)

## Récupérer les clients avec les commandes - variations sur un thème

Cela permettra d'obtenir toutes les commandes pour tous les clients:

```

SELECT c.CustomerName, o.OrderID
  FROM Customers AS c
 INNER JOIN Orders AS o
    ON c.CustomerID = o.CustomerID
 ORDER BY c.CustomerName, o.OrderID;

```

Cela comptera le nombre de commandes pour chaque client:

```

SELECT c.CustomerName, COUNT(*) AS 'Order Count'
  FROM Customers AS c
 INNER JOIN Orders AS o
    ON c.CustomerID = o.CustomerID
 GROUP BY c.CustomerID;
 ORDER BY c.CustomerName;

```

Aussi, compte, mais probablement plus vite:

```

SELECT  c.CustomerName,
        ( SELECT COUNT(*) FROM Orders WHERE CustomerID = c.CustomerID ) AS 'Order Count'
FROM Customers AS c
ORDER BY c.CustomerName;

```

N'indiquez que le client avec les commandes.

```

SELECT  c.CustomerName,
FROM Customers AS c
WHERE EXISTS ( SELECT * FROM Orders WHERE CustomerID = c.CustomerID )
ORDER BY c.CustomerName;

```

## Full Outer Join

MySQL ne prend pas en charge `FULL OUTER JOIN`, mais il existe des moyens pour en émuler un.

## Configuration des données

```

-- -----
-- Table structure for `owners`
-- -----
DROP TABLE IF EXISTS `owners`;
CREATE TABLE `owners` (
  `owner_id` int(11) NOT NULL AUTO_INCREMENT,
  `owner` varchar(30) DEFAULT NULL,
  PRIMARY KEY (`owner_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=latin1;

-- -----
-- Records of owners
-- -----
INSERT INTO `owners` VALUES ('1', 'Ben');
INSERT INTO `owners` VALUES ('2', 'Jim');
INSERT INTO `owners` VALUES ('3', 'Harry');
INSERT INTO `owners` VALUES ('6', 'John');
INSERT INTO `owners` VALUES ('9', 'Ellie');

-- -----
-- Table structure for `tools`
-- -----
DROP TABLE IF EXISTS `tools`;
CREATE TABLE `tools` (
  `tool_id` int(11) NOT NULL AUTO_INCREMENT,
  `tool` varchar(30) DEFAULT NULL,
  `owner_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`tool_id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=latin1;

-- -----
-- Records of tools
-- -----
INSERT INTO `tools` VALUES ('1', 'Hammer', '9');
INSERT INTO `tools` VALUES ('2', 'Pliers', '1');
INSERT INTO `tools` VALUES ('3', 'Knife', '1');
INSERT INTO `tools` VALUES ('4', 'Chisel', '2');
INSERT INTO `tools` VALUES ('5', 'Hacksaw', '1');
INSERT INTO `tools` VALUES ('6', 'Level', null);
INSERT INTO `tools` VALUES ('7', 'Wrench', null);
INSERT INTO `tools` VALUES ('8', 'Tape Measure', '9');

```

```
INSERT INTO `tools` VALUES ('9', 'Screwdriver', null);
INSERT INTO `tools` VALUES ('10', 'Clamp', null);
```

## Que voulons-nous voir?

Nous voulons obtenir une liste dans laquelle nous voyons qui possède quels outils et quels outils peuvent ne pas avoir de propriétaire.

## Les requêtes

Pour ce faire, nous pouvons combiner deux requêtes en utilisant `UNION`. Dans cette première requête, nous rejoignons les outils sur les propriétaires en utilisant un `LEFT JOIN`. Cela va ajouter tous nos propriétaires à notre jeu de résultats, peu importe s'ils possèdent réellement des outils.

Dans la deuxième requête, nous utilisons `RIGHT JOIN` pour joindre les outils aux propriétaires. De cette manière, nous parvenons à obtenir tous les outils de notre jeu de résultats. Si leur propriétaire ne leur appartient pas, leur colonne propriétaire contiendra simplement `NULL`. En ajoutant une `WHERE` qui filtre par `owners.owner_id IS NULL` nous définissons le résultat comme les jeux de données, qui n'ont pas déjà été renvoyés par la première requête, car nous ne recherchons que les données dans la table jointe droite.

Puisque nous utilisons `UNION ALL` le jeu de résultats de la deuxième requête sera attaché au premier ensemble de résultats des requêtes.

```
SELECT `owners`.`owner`, tools.tool
FROM `owners`
LEFT JOIN `tools` ON `owners`.`owner_id` = `tools`.`owner_id`
UNION ALL
SELECT `owners`.`owner`, tools.tool
FROM `owners`
RIGHT JOIN `tools` ON `owners`.`owner_id` = `tools`.`owner_id`
WHERE `owners`.`owner_id` IS NULL;
```

```
+-----+-----+
| owner | tool      |
+-----+-----+
| Ben   | Pliers    |
| Ben   | Knife     |
| Ben   | Hacksaw   |
| Jim   | Chisel    |
| Harry | NULL      |
| John  | NULL      |
| Ellie | Hammer   |
| Ellie | Tape Measure |
| NULL  | Level     |
| NULL  | Wrench    |
| NULL  | Screwdriver |
| NULL  | Clamp     |
+-----+-----+
12 rows in set (0.00 sec)
```

## Inner-join pour 3 tables

Supposons que nous ayons trois tables qui peuvent être utilisées pour des sites Web simples

avec des balises.

- La table de ping est pour les postes.
- Deuxième pour les balises
- Troisième pour la relation Tags & Post

table de ping "jeu vidéo"

| id | Titre             | reg_date   | Contenu |
|----|-------------------|------------|---------|
| 1  | Bioshock Infinite | 2016-08-08 | ....    |

table "tags"

| id | prénom    |
|----|-----------|
| 1  | yennefer  |
| 2  | Elizabeth |

Table "tags\_meta"

| post_id | tag_id |
|---------|--------|
| 1       | 2      |

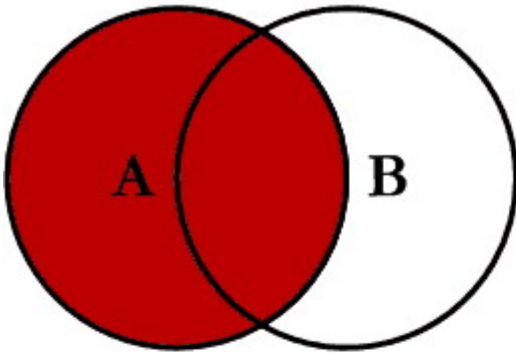
```
SELECT videogame.id,  
       videogame.title,  
       videogame.reg_date,  
       tags.name,  
       tags_meta.post_id  
FROM tags_meta  
INNER JOIN videogame ON videogame.id = tags_meta.post_id  
INNER JOIN tags ON tags.id = tags_meta.tag_id  
WHERE tags.name = "elizabeth"  
ORDER BY videogame.reg_date
```

ce code peut renvoyer tous les messages relatifs à cette balise "#elizabeth"

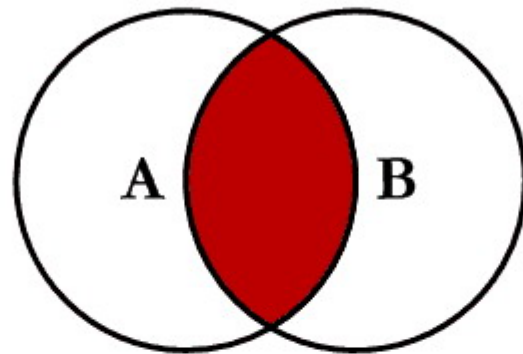
## Jointure visualisée

Si vous êtes une personne visuellement orientée, ce diagramme de Venn peut vous aider à comprendre les différents types de `JOIN` qui existent dans MySQL.

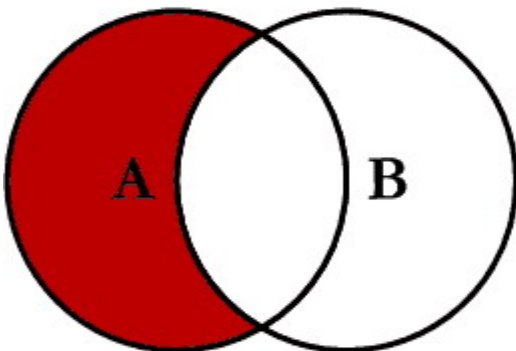
# SQL JOINS



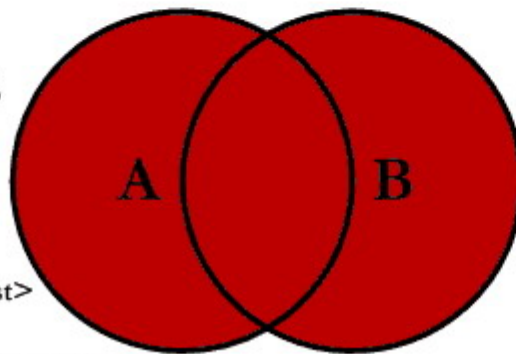
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



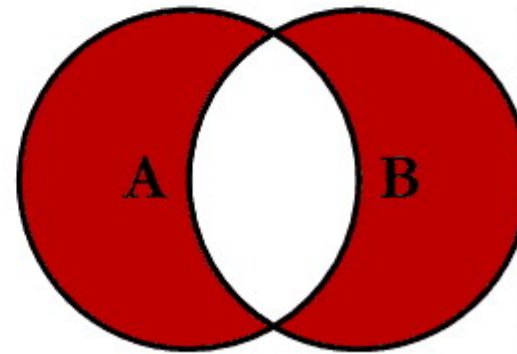
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



© C.L. Moffatt, 2008

Lire Joint en ligne: <https://riptutorial.com/fr/mysql/topic/2736/joint>

---

# Chapitre 32: JSON

## Introduction

Depuis MySQL 5.7.8, MySQL supporte un type de données JSON natif qui permet un accès efficace aux données dans les documents JSON (JavaScript Object Notation).

<https://dev.mysql.com/doc/refman/5.7/fr/json.html>

## Remarques

Depuis MySQL 5.7.8, MySQL est livré avec un type JSON. Beaucoup de développeurs ont enregistré des données JSON dans des colonnes de texte pour une heure de journalisation mais le type JSON est différent, les données sont enregistrées au format binaire après validation. Cela évite d'avoir à analyser le texte à chaque lecture.

## Exemples

### Créer un tableau simple avec une clé primaire et un champ JSON

```
CREATE TABLE table_name (  
    id INT NOT NULL AUTO_INCREMENT,  
    json_col JSON,  
    PRIMARY KEY(id)  
);
```

### Insérer un simple JSON

```
INSERT INTO  
    table_name (json_col)  
VALUES  
    ('{"City": "Galle", "Description": "Best damn city in the world"}');
```

C'est simple car il peut être obtenu, mais notez que les clés de dictionnaire JSON doivent être entourées de guillemets doubles. Si la requête réussit, les données seront stockées dans un format binaire.

### Insérer des données mixtes dans un champ JSON.

Cela insère un dictionnaire json où l'un des membres est un tableau de chaînes dans la table créée dans un autre exemple.

```
INSERT INTO myjson(dict)  
VALUES ('{"opening": "Sicilian", "variations": ["pelikan", "dragon", "najdorf"]}');
```

Notez, encore une fois, que vous devez faire attention à l'utilisation de guillemets simples et



doubles. Le tout doit être emballé dans des guillemets simples.

## Mise à jour d'un champ JSON

Dans l'exemple précédent, nous avons vu comment des types de données mixtes peuvent être insérés dans un champ JSON. Que faire si nous voulons mettre à jour ce champ? Nous allons ajouter *scheveningen* au tableau nommé `variations` dans l'exemple précédent.

```
UPDATE
  myjson
SET
  dict=JSON_ARRAY_APPEND(dict, '$.variations', 'scheveningen')
WHERE
  id = 2;
```

Remarques:

1. Le tableau `$.variations` dans notre dictionnaire json. Le symbole `$` représente la documentation json. Pour une explication complète des chemins json reconnus par mysql, reportez-vous à <https://dev.mysql.com/doc/refman/5.7/en/json-path-syntax.html>
2. Comme nous n'avons pas encore d'exemple sur l'interrogation de champs json, cet exemple utilise la clé primaire.

Maintenant, si nous faisons `SELECT * FROM myjson` nous verrons

```
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-+
| id | dict
|
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+
| 2  | {"opening": "Sicilian", "variations": ["pelikan", "dragon", "najdorf", "scheveningen"]}
|
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-+
1 row in set (0.00 sec)
```

## Données CAST au type JSON

Cela convertit les chaînes json valides en type JSON MySQL:

```
SELECT CAST('[1,2,3]' as JSON) ;
SELECT CAST('{"opening": "Sicilian", "variations": ["pelikan", "dragon", "najdorf"]}' as JSON);
```

## Créer un objet et un tableau Json

`JSON_OBJECT` crée des objets JSON:

```
SELECT JSON_OBJECT('key1', col1 , 'key2', col2 , 'key3', 'col3') as myobj;
```

JSON\_ARRAY crée également JSON Array:

```
SELECT JSON_ARRAY(col1,col2,'col3') as myarray;
```

Note: myobj.key3 et myarray [2] sont "col3" en tant que chaîne fixe.

Données mixtes JSON également:

```
SELECT JSON_OBJECT("opening","Sicilian",  
"variations",JSON_ARRAY("pelikan","dragon","najdorf") ) as mymixed ;
```

Lire JSON en ligne: <https://riptutorial.com/fr/mysql/topic/2985/json>

---

# Chapitre 33: L'optimisation des performances

## Syntaxe

- N'utilisez pas DISTINCT et GROUP BY dans le même SELECT.
- Ne paginez pas avec OFFSET, "rappelez-vous où vous vous êtes arrêté".
- WHERE (a, b) = (22,33) n'optimise pas du tout.
- Dites explicitement ALL ou DISTINCT après UNION - cela vous rappelle de choisir entre le ALL plus rapide ou le DISTINCT plus lent.
- N'utilisez pas SELECT \*, surtout si vous n'avez pas besoin de colonnes TEXT ou BLOB. Il y a une surcharge dans les tables tmp et la transmission.
- C'est plus rapide lorsque GROUP BY et ORDER BY peuvent avoir exactement la même liste.
- N'utilisez pas FORCE INDEX; ça peut aider aujourd'hui, mais ça va probablement faire mal demain.

## Remarques

Voir aussi les discussions sur ORDER BY, LIKE, REGEXP, etc.

[Livre de recettes sur la création d'index optimaux](#) .

## Exemples

### Ajouter le bon index

C'est un sujet énorme, mais c'est aussi le plus important problème de "performance".

La principale leçon pour un novice est d'apprendre des index "composites". Voici un exemple rapide:

```
INDEX(last_name, first_name)
```

est excellent pour ceux-ci:

```
WHERE last_name = '...'  
WHERE first_name = '...' AND last_name = '...' -- (order in WHERE does not matter)
```

mais pas pour

```
WHERE first_name = '...' -- order in INDEX _does_ matter
WHERE last_name = '...' OR first_name = '...' -- "OR" is a killer
```

## Définissez le cache correctement

`innodb_buffer_pool_size` devrait représenter environ 70% de la mémoire RAM disponible.

## Évitez les constructions inefficaces

```
x IN ( SELECT ... )
```

se transformer en un `JOIN`

Si possible, évitez `OR`.

Ne «cache» pas une colonne indexée dans une fonction, telle que `WHERE DATE(x) = ...`; reformuler comme `WHERE x = ...`.

Vous pouvez généralement éviter `WHERE LCASE(name1) = LCASE(name2)` en ayant un classement approprié.

Ne pas utiliser `OFFSET` pour la "pagination", au lieu de cela "rappelez-vous où vous vous êtes arrêté".

Évitez `SELECT * ...` (à moins que le débogage).

*Note à Maria Deleva, Barranka, Batsu: Ceci est un espace réservé; veuillez supprimer ces éléments lorsque vous construisez des exemples à grande échelle. Une fois que vous avez fait ceux que vous pouvez, je vais aller de l'avant pour élaborer le reste et / ou les lancer.*

## Les négatifs

Voici certaines choses qui ne sont pas susceptibles d'aider les performances. Ils proviennent d'informations obsolètes et / ou de naïveté.

- InnoDB s'est amélioré au point où il est peu probable que MyISAM soit meilleur.
- `PARTITIONing` offre rarement des avantages de performance; cela peut même nuire à la performance.
- La définition de `query_cache_size` supérieure à 100M *nuira* généralement aux performances.
- L'augmentation du nombre de valeurs dans `my.cnf` peut conduire à un «échange», ce qui constitue un *grave* problème de performance.
- Les "index de préfixe" (tels que `INDEX(foo(20))`) sont généralement inutiles.
- `OPTIMIZE TABLE` est presque toujours inutile. (Et cela implique de verrouiller la table.)

## Avoir un INDEX

La chose la plus importante pour accélérer une requête sur une table non minuscule est d'avoir un index approprié.

```
WHERE a = 12 --> INDEX(a)
WHERE a > 12 --> INDEX(a)

WHERE a = 12 AND b > 78 --> INDEX(a,b) is more useful than INDEX(b,a)
WHERE a > 12 AND b > 78 --> INDEX(a) or INDEX(b); no way to handle both ranges

ORDER BY x --> INDEX(x)
ORDER BY x, y --> INDEX(x,y) in that order
ORDER BY x DESC, y ASC --> No index helps - because of mixing ASC and DESC
```

## Ne cache pas dans la fonction

Une erreur courante consiste à masquer une colonne indexée dans un appel de fonction. Par exemple, cela ne peut être aidé par un index:

```
WHERE DATE(dt) = '2000-01-01'
```

Au lieu de cela, `INDEX(dt)` peut alors utiliser l'index:

```
WHERE dt = '2000-01-01' -- if `dt` is datatype `DATE`
```

Cela fonctionne pour `DATE`, `DATETIME`, `TIMESTAMP` et même `DATETIME(6)` (microsecondes):

```
WHERE dt >= '2000-01-01'
AND dt < '2000-01-01' + INTERVAL 1 DAY
```

## OU

En général `OR` tue l'optimisation.

```
WHERE a = 12 OR b = 78
```

ne peut pas utiliser `INDEX(a,b)`, et peut ou non utiliser `INDEX(a)`, `INDEX(b)` via "index merge". La fusion d'index est meilleure que rien, mais seulement à peine.

```
WHERE x = 3 OR x = 5
```

est transformé en

```
WHERE x IN (3, 5)
```

qui *peut* utiliser un index avec `x` dedans.

## Sous-requêtes

Les sous-requêtes existent en plusieurs versions et ont un potentiel d'optimisation différent. Tout d'abord, notez que les sous-requêtes peuvent être "corrélées" ou "non corrélées". Corrélé signifie qu'ils dépendent d'une valeur extérieure à la sous-requête. Cela implique généralement que la

sous-requête *doit* être réévaluée pour chaque valeur externe.

Cette sous-requête corrélée est souvent très bonne. Note: il doit retourner au plus 1 valeur. Il est souvent utile comme alternative à, mais pas nécessairement plus rapide que, un `LEFT JOIN`.

```
SELECT a, b, ( SELECT ... FROM t WHERE t.x = u.x ) AS c
  FROM u ...
SELECT a, b, ( SELECT MAX(x) ... ) AS c
  FROM u ...
SELECT a, b, ( SELECT x FROM t ORDER BY ... LIMIT 1 ) AS c
  FROM u ...
```

Ceci est généralement non corrélé:

```
SELECT ...
  FROM ( SELECT ... ) AS a
 JOIN b ON ...
```

Notes sur le `FROM-SELECT` :

- Si elle retourne 1 ligne, super.
- Un bon paradigme (encore une fois "1 row") est que la sous-requête soit ( `SELECT @n := 0` ), initialisant ainsi une variable `@ @` à utiliser dans le reste ou dans la requête.
- Si elle renvoie de nombreuses lignes *et que* le `JOIN` est également ( `SELECT ...` ) avec plusieurs lignes, l'efficacité peut être terrible. Avant le 5.6, il n'y avait pas d'index, donc c'est devenu un `CROSS JOIN` ; 5.6+ consiste à déduire le meilleur index sur les tables temporaires, puis à le générer, pour le jeter ensuite une fois terminé avec `SELECT`.

## JOIN + GROUP BY

Un problème commun qui conduit à une requête inefficace est quelque chose comme ceci:

```
SELECT ...
  FROM a
 JOIN b ON ...
 WHERE ...
 GROUP BY a.id
```

Tout d'abord, le `JOIN` développe le nombre de lignes. puis le `GROUP BY` réduit le nombre de lignes dans `a`.

Il n'y a peut-être pas de bons choix pour résoudre ce problème d'implosion. Une option possible consiste à transformer `JOIN` en une sous-requête corrélée dans `SELECT`. Cela élimine également le `GROUP BY`.

Lire [L'optimisation des performances en ligne](https://riptutorial.com/fr/mysql/topic/4292/-optimisation-des-performances): <https://riptutorial.com/fr/mysql/topic/4292/-optimisation-des-performances>

---

# Chapitre 34: Limite et décalage

## Syntaxe

- `SELECT column_1 [, column_2]`  
`FROM table_1`  
`ORDER BY order_column`  
`LIMIT row_count [OFFSET row_offset]`
- `SELECT column_1 [, column_2]`  
`FROM table_1`  
`ORDER BY order_column`  
`LIMIT [row_offset,] row_count`

## Remarques

"Limite" pourrait signifier "Nombre maximum de lignes dans une table".

"Décalage" signifie le choix du numéro de `row` (à ne pas confondre avec la valeur de la clé primaire ou une valeur de champ quelconque)

## Exemples

### Relation Limite et Offset

Considérant la table d' `users` suivante:

| id | Nom d'utilisateur |
|----|-------------------|
| 1  | Utilisateur1      |
| 2  | Utilisateur2      |
| 3  | Utilisateur3      |
| 4  | Utilisateur4      |
| 5  | Utilisateur5      |

Afin de contraindre le nombre de lignes dans le jeu de résultats d'une [requête SELECT](#), la clause `LIMIT` peut être utilisée avec un ou deux entiers positifs comme arguments (zéro inclus).

---

## Clause `LIMIT` avec un argument

Lorsqu'un argument est utilisé, le jeu de résultats sera uniquement limité au nombre spécifié de la

manière suivante:

```
SELECT * FROM users ORDER BY id ASC LIMIT 2
```

| id | Nom d'utilisateur |
|----|-------------------|
| 1  | Utilisateur1      |
| 2  | Utilisateur2      |

Si la valeur de l'argument est 0 , le jeu de résultats sera vide.

Notez également que la clause `ORDER BY` peut être importante pour spécifier les premières lignes du jeu de résultats qui sera présenté (lors du classement par une autre colonne).

## Clause `LIMIT` avec deux arguments

Lorsque deux arguments sont utilisés dans une clause `LIMIT` :

- le **premier** argument représente la ligne à partir de laquelle les lignes du jeu de résultats seront présentées - ce nombre est souvent mentionné comme un **décalage** , car il représente la ligne précédant la ligne initiale du jeu de résultats contraint. Cela permet à l'argument de recevoir 0 comme valeur et de prendre ainsi en compte la première ligne du jeu de résultats non contraint.
- le **second** argument spécifie le nombre maximal de lignes à retourner dans le jeu de résultats (comme dans l'exemple d'un argument).

Par conséquent, la requête:

```
SELECT * FROM users ORDER BY id ASC LIMIT 2, 3
```

Présente le jeu de résultats suivant:

| id | Nom d'utilisateur |
|----|-------------------|
| 3  | Utilisateur3      |
| 4  | Utilisateur4      |
| 5  | Utilisateur5      |

Notez que lorsque l'argument **offset** est 0 , le jeu de résultats sera équivalent à une clause `LIMIT` à un argument. Cela signifie que les 2 requêtes suivantes:

```
SELECT * FROM users ORDER BY id ASC LIMIT 0, 2
```

```
SELECT * FROM users ORDER BY id ASC LIMIT 2
```



Produire le même jeu de résultats:

| id | Nom d'utilisateur |
|----|-------------------|
| 1  | Utilisateur1      |
| 2  | Utilisateur2      |

## Mot clé `OFFSET` : syntaxe alternative

Une syntaxe alternative pour la clause `LIMIT` avec deux arguments consiste à utiliser le mot clé `OFFSET` après le premier argument de la manière suivante:

```
SELECT * FROM users ORDER BY id ASC LIMIT 2 OFFSET 3
```

Cette requête renvoie le jeu de résultats suivant:

| id | Nom d'utilisateur |
|----|-------------------|
| 3  | Utilisateur3      |
| 4  | Utilisateur4      |

Notez que dans cette syntaxe alternative les arguments ont leurs positions changées:

- le **premier** argument représente le nombre de lignes à renvoyer dans le jeu de résultats;
- le **second** argument représente le décalage.

Lire Limite et décalage en ligne: <https://riptutorial.com/fr/mysql/topic/548/limite-et-decalage>

---

# Chapitre 35: Manipulation des fuseaux horaires

## Remarques

Lorsque vous devez gérer les informations de temps pour une base d'utilisateurs mondiale dans MySQL, utilisez le type de données `TIMESTAMP` dans vos tables.

Pour chaque utilisateur, stockez une colonne de fuseau horaire de préférence utilisateur. `VARCHAR (64)` est un bon type de données pour cette colonne. Lorsqu'un utilisateur s'inscrit pour utiliser votre système, demandez la valeur du fuseau horaire. Le mien est Atlantic Time, `America/Edmonton` . Vous pourriez ou non être en `Asia/Kolkata` ou en `Australia/NSW` . Pour une interface utilisateur pour ce paramètre de préférence utilisateur, le logiciel WordPress.org a un bon exemple.

Enfin, chaque fois que vous établissez une connexion depuis votre programme hôte (Java, php, etc.) vers votre SGBD au nom d'un utilisateur, émettez la commande SQL.

```
SET SESSION time_zone='(whatever tz string the user gave you)'
```

avant de manipuler des données d'utilisateur impliquant des temps. Ensuite, tous les temps `TIMESTAMP` vous avez installés seront affichés à l'heure locale de l'utilisateur.

Cela entraînera la conversion de tous les temps dans vos tables en UTC et la conversion de toutes les heures en données locales. Cela fonctionne correctement pour `NOW ()` et `CURDATE ()`. Encore une fois, vous devez utiliser `TIMESTAMP` et pas les types de données `DATETIME` ou `DATE` pour cela.

Assurez-vous que le système d'exploitation de votre serveur et les fuseaux horaires par défaut de MySQL sont définis sur UTC. Si vous ne le faites pas avant de charger des informations dans votre base de données, il sera presque impossible de les corriger. Si vous utilisez un fournisseur pour exécuter MySQL, insistez pour que ce soit le cas.

## Exemples

**Récupérez la date et l'heure actuelles dans un fuseau horaire particulier.**

Cela récupère la valeur de `NOW ()` en heure locale, en heure standard en Inde, puis à nouveau en UTC.

```
SELECT NOW();
SET time_zone='Asia/Kolkata';
SELECT NOW();
SET time_zone='UTC';
SELECT NOW();
```

## Convertir une valeur stockée `DATE` ou `DATETIME` dans un autre fuseau horaire.

Si vous avez un `DATE` ou `DATETIME` stocké (dans une colonne quelque part), il a été stocké par rapport à un fuseau horaire, mais en MySQL, le fuseau horaire n'est pas stocké avec la valeur. Donc, si vous voulez le convertir en un autre fuseau horaire, vous pouvez le faire, mais vous devez connaître le fuseau horaire d'origine. Utiliser `CONVERT_TZ()` effectue la conversion. Cet exemple montre les lignes vendues en Californie à l'heure locale.

```
SELECT CONVERT_TZ(date_sold, 'UTC', 'America/Los_Angeles') date_sold_local
FROM sales
WHERE state_sold = 'CA'
```

## Récupère les valeurs stockées de `TIMESTAMP` dans un fuseau horaire particulier

C'est vraiment facile. Toutes les valeurs `TIMESTAMP` sont stockées dans le temps universel et sont toujours converties au paramètre `time_zone` actuel à chaque rendu.

```
SET SESSION time_zone='America/Los_Angeles';
SELECT timestamp_sold
FROM sales
WHERE state_sold = 'CA'
```

Pourquoi est-ce? `TIMESTAMP` valeurs `TIMESTAMP` sont basées sur le [type de données](#) vénérable `UNIX time_t`. Ces horodatages UNIX sont stockés sous forme de nombre de secondes depuis le 1970-01-01 00:00:00 UTC.

**Remarquez que les valeurs `TIMESTAMP` sont stockées dans le temps universel. `DATE` valeurs `DATE` et `DATETIME` sont stockées quelle que soit l'heure locale au moment de leur stockage.**

## Quel est le paramètre de fuseau horaire local de mon serveur?

Chaque serveur a un paramètre global `time_zone` par défaut, configuré par le propriétaire de la machine serveur. Vous pouvez trouver le réglage actuel du fuseau horaire de cette manière:

```
SELECT @@time_zone
```

Malheureusement, cela donne généralement la valeur `SYSTEM`, ce qui signifie que le temps MySQL est régi par le paramètre de fuseau horaire du système d'exploitation du serveur.

Cette séquence de requêtes (oui, [c'est un hack](#)) vous redonne le décalage en minutes entre le paramètre de fuseau horaire du serveur et UTC.

```
CREATE TEMPORARY TABLE times (dt DATETIME, ts TIMESTAMP);
SET time_zone = 'UTC';
INSERT INTO times VALUES (NOW(), NOW());
SET time_zone = 'SYSTEM';
```

```
SELECT dt, ts, TIMESTAMPDIFF(MINUTE, dt, ts)offset FROM times;
DROP TEMPORARY TABLE times;
```

Comment cela marche-t-il? Les deux colonnes de la table temporaire avec différents types de données constituent l'indice. Les types de données `DATETIME` sont toujours stockés dans l'heure locale dans les tables et les `TIMESTAMP` dans l'UTC. Ainsi, l'instruction `INSERT`, exécutée lorsque le paramètre `time_zone` est défini sur UTC, stocke deux valeurs de date / heure identiques.

Ensuite, l'instruction `SELECT` est effectuée lorsque `time_zone` est défini sur l'heure locale du serveur. `TIMESTAMP` s sont toujours traduits de leur forme UTC stockée en heure locale dans les instructions `SELECT`. `DATETIME` ne sont pas. Ainsi, l'opération `TIMESTAMPDIFF(MINUTE...)` calcule la différence entre l'heure locale et l'heure universelle.

## Quelles valeurs `time_zone` sont disponibles sur mon serveur?

Pour obtenir une liste des valeurs `time_zone` possibles dans votre instance de serveur MySQL, utilisez cette commande.

```
SELECT mysql.time_zone_name.name
```

Normalement, cela montre la [liste ZoneInfo des fuseaux horaires](#) maintenue par Paul Eggert à la [Internet Assigned Numbers Authority](#). Dans le monde entier, il y a environ 600 fuseaux horaires.

Les systèmes d'exploitation de type Unix (distributions Linux, distributions BSD et distributions Mac OS modernes, par exemple) reçoivent des mises à jour de routine. L'installation de ces mises à jour sur un système d'exploitation permet aux instances MySQL en cours d'exécution de suivre les changements de fuseau horaire et les changements d'heure d'été / heure standard.

Si vous obtenez une liste beaucoup plus courte de noms de fuseaux horaires, votre serveur est soit mal configuré, soit exécuté sous Windows. [Voici les instructions](#) pour votre administrateur de serveur pour installer et maintenir la liste ZoneInfo.

Lire Manipulation des fuseaux horaires en ligne:

<https://riptutorial.com/fr/mysql/topic/7849/manipulation-des-fuseaux-horaires>

# Chapitre 36: METTRE À JOUR

## Syntaxe

- UPDATE [LOW\_PRIORITY] [IGNORE] nomTable SET column1 = expression1, column2 = expression2, ... [conditions WHERE]; // Mise à jour simple d'une table unique
- UPDATE [LOW\_PRIORITY] [IGNORE] nomTable SET column1 = expression1, column2 = expression2, ... [conditions WHERE] [expression ORDER BY [ASC | DESC]] [LIMIT row\_count]; // Mise à jour avec ordre par et limite
- UPDATE [LOW\_PRIORITY] [IGNORE] table1, table2, ... SET column1 = expression1, column2 = expression2, ... [conditions WHERE]; // Mise à jour de plusieurs tables

## Exemples

### Mise à jour de base

### Mise à jour d' une ligne

```
UPDATE customers SET email='luke_smith@email.com' WHERE id=1
```

Cette requête met à jour le contenu du `email` dans la table `customers` à la chaîne `luke_smith@email.com` où la valeur de `id` est égale à 1. Les anciens et les nouveaux contenus de la table de base de données sont illustrés ci-dessous respectivement à gauche et à droite:

| customers |           |          |                  |
|-----------|-----------|----------|------------------|
| id        | firstname | lastname | email            |
| 1         | Luke      | Smith    | luke@example.com |
| 2         | Anna      | Carey    | anna@example.com |
| 3         | Todd      | Winters  | todd@example.com |

| customers |           |          |                      |
|-----------|-----------|----------|----------------------|
| id        | firstname | lastname | email                |
| 1         | Luke      | Smith    | luke_smith@email.com |
| 2         | Anna      | Carey    | anna@example.com     |
| 3         | Todd      | Winters  | todd@example.com     |

### Mise à jour de toutes les lignes

```
UPDATE customers SET lastname='smith'
```

Cette requête met à jour le contenu du `lastname` pour chaque entrée de la table `customers`. L'ancien et le nouveau contenu de la table de base de données sont illustrés ci-dessous, respectivement à gauche et à droite:

| customers |           |          |                  |
|-----------|-----------|----------|------------------|
| id        | firstname | lastname | email            |
| 1         | Luke      | Smith    | luke@example.com |
| 2         | Anna      | Carey    | anna@example.com |
| 3         | Todd      | Winters  | todd@example.com |

| customers |           |          |                  |
|-----------|-----------|----------|------------------|
| id        | firstname | lastname | email            |
| 1         | Luke      | Smith    | luke@example.com |
| 2         | Anna      | Smith    | anna@example.com |
| 3         | Todd      | Smith    | todd@example.com |

**Remarque:** il est nécessaire d'utiliser des clauses conditionnelles (WHERE) dans la requête UPDATE. Si vous n'utilisez pas de clause conditionnelle, tous les enregistrements de cet attribut seront mis à jour. Dans l'exemple ci-dessus, la nouvelle valeur (Smith) du nom de famille dans la table clients est définie sur toutes les lignes.

## Mise à jour avec le modèle de jointure

Considérons une table de production appelée `questions_mysql` et une table `iwtQuestions` (table de travail importée) représentant le dernier lot de données CSV importées à partir d'un [LOAD DATA INFILE](#). La table de travail est tronquée avant l'importation, les données sont importées et ce processus n'est pas affiché ici.

Mettez à jour nos données de production en utilisant une jointure à nos données de table de travail importées.

```
UPDATE questions_mysql q -- our real table for production
join iwtQuestions i -- imported worktable
ON i.qId = q.qId
SET q.closeVotes = i.closeVotes,
q.votes = i.votes,
q.answers = i.answers,
q.views = i.views;
```

Les alias `q` et `i` sont utilisés pour abrégier les références de la table. Cela facilite le développement et la lisibilité.

`qId`, la clé primaire, représente l'identifiant de la question Stackoverflow. Quatre colonnes sont mises à jour pour faire correspondre les lignes de la jointure.

## MISE À JOUR avec ORDER BY et LIMIT

Si la clause `ORDER BY` est spécifiée dans votre instruction SQL update, les lignes sont mises à jour dans l'ordre spécifié.

Si la clause `LIMIT` est spécifiée dans votre instruction SQL, cela limite le nombre de lignes pouvant être mises à jour. Il n'y a pas de limite, si la clause `LIMIT` n'est pas spécifiée.

`ORDER BY` et `LIMIT` ne peuvent pas être utilisés pour la mise à jour multi-tables.

Syntaxe pour MySQL UPDATE avec `ORDER BY` et `LIMIT` est,

```
UPDATE [ LOW_PRIORITY ] [ IGNORE ]
tableName
SET column1 = expression1,
```

```

    column2 = expression2,
    ...
[WHERE conditions]
[ORDER BY expression [ ASC | DESC ]]
[LIMIT row_count];

---> Example
UPDATE employees SET isConfirmed=1 ORDER BY joiningDate LIMIT 10

```

Dans l'exemple ci-dessus, 10 lignes seront mises à jour en fonction de l'ordre des employés `joiningDate`.

## MISE À JOUR de table multiple

Dans `UPDATE` tables multiples, il met à jour les lignes de chaque table spécifiée qui satisfont aux conditions. Chaque ligne correspondante est mise à jour une fois, même si elle correspond aux conditions plusieurs fois.

Dans les tables multiples `UPDATE`, `ORDER BY` et `LIMIT` ne peuvent pas être utilisés.

La syntaxe pour la table multi `UPDATE` est,

```

UPDATE [LOW_PRIORITY] [IGNORE]
table1, table2, ...
    SET column1 = expression1,
        column2 = expression2,
        ...
[WHERE conditions]

```

Prenons par exemple deux tables, `products` et `salesOrders`. Dans le cas où nous réduisons la quantité d'un produit particulier de la commande client qui est déjà passée. Ensuite, nous devons également augmenter cette quantité dans notre colonne de stock de `products`. Cela peut être fait dans une seule déclaration de mise à jour SQL comme ci-dessous.

```

UPDATE products, salesOrders
    SET salesOrders.Quantity = salesOrders.Quantity - 5,
        products.availableStock = products.availableStock + 5
WHERE products.productId = salesOrders.productId
    AND salesOrders.orderId = 100 AND salesOrders.productId = 20;

```

Dans l'exemple ci-dessus, la quantité '5' sera réduite à partir de la table `salesOrders` et la même chose sera augmentée dans la table des `products` fonction des conditions `WHERE`.

## MISE À JOUR en vrac

Lors de la mise à jour de plusieurs lignes avec des valeurs différentes, il est beaucoup plus rapide d'utiliser une mise à jour groupée.

```

UPDATE people
SET name =
    (CASE id WHEN 1 THEN 'Karl'
         WHEN 2 THEN 'Tom'

```

```
        WHEN 3 THEN 'Mary'  
    END)  
WHERE id IN (1,2,3);
```

En mettant à jour en bloc, une seule requête peut être envoyée au serveur au lieu d'une requête pour chaque ligne à mettre à jour. Les cas doivent contenir tous les paramètres possibles recherchés dans la clause `WHERE` .

Lire **METTRE À JOUR** en ligne: <https://riptutorial.com/fr/mysql/topic/2738/mettre-a-jour>



---

# Chapitre 37: Moteur MyISAM

## Remarques

Au fil des ans, InnoDB s'est amélioré au point où il est presque toujours meilleur que MyISAM, du moins les versions actuellement prises en charge. Bottom line: N'utilisez pas MyISAM, sauf peut-être pour les tables qui sont vraiment temporaires.

Un avantage de MyISAM sur InnoDB reste: il est de 2 x 3 fois plus petit en espace disque.

Quand InnoDB est sorti pour la première fois, MyISAM était toujours un moteur viable. Mais avec l'arrivée de XtraDB et 5.6, InnoDB est devenu "meilleur" que MyISAM dans la plupart des tests.

La rumeur veut que la prochaine version majeure élimine le besoin de MyISAM en créant des tables InnoDB véritablement temporaires et en déplaçant les tables système dans InnoDB.

## Exemples

### MOTEUR = MyISAM

```
CREATE TABLE foo (  
    ...  
) ENGINE=MyISAM;
```

Lire Moteur MyISAM en ligne: <https://riptutorial.com/fr/mysql/topic/4710/moteur-myisam>

---

# Chapitre 38: Mots réservés

## Introduction

MySQL a des noms spéciaux appelés *mots réservés*. Un mot réservé peut être utilisé comme identifiant pour une table, une colonne, etc. seulement s'il est encapsulé dans des backticks (`), sinon cela entraînera une erreur.

Pour éviter de telles erreurs, n'utilisez pas de mots réservés en tant qu'identificateurs ou enveloppez l'identifiant incriminé dans des raccourcis.

## Remarques

Vous trouverez ci-dessous tous les mots réservés (de [la documentation officielle](#)):

- ACCESSIBLE
- AJOUTER
- TOUT
- MODIFIER
- ANALYSER
- ET
- COMME
- ASC
- ASENSITIVE
- AVANT
- ENTRE
- BIGINT
- BINAIRE
- GOUTTE
- TOUS LES DEUX
- PAR
- APPEL
- CASCADE
- CAS
- CHANGEMENT
- CARBONISER
- PERSONNAGE
- VÉRIFIER
- COLLATIONNER
- COLONNE
- CONDITION
- CONTRAINTE
- CONTINUER
- CONVERTIR
- CRÉER

- TRAVERSER
- DATE ACTUELLE
- HEURE ACTUELLE
- CURRENT\_TIMESTamp
- UTILISATEUR ACTUEL
- LE CURSEUR
- BASE DE DONNÉES
- BASES DE DONNÉES
- JOUR\_HOUR
- JOUR\_MICROSECOND
- DAY\_MINUTE
- DAY\_SECOND
- DÉC
- DÉCIMAL
- DÉCLARER
- DÉFAUT
- DIFFÉRÉ
- EFFACER
- DESC
- DÉCRIRE
- DÉTERMINISTIQUE
- DISTINCT
- DISTINCTROW
- DIV
- DOUBLE
- LAISSEZ TOMBER
- DOUBLE
- CHAQUE
- AUTRE
- ELSEIF
- ENFERMÉ
- ÉCHAPPÉ
- EXISTE
- SORTIE
- EXPLIQUE
- FAUX
- FETCH
- FLOTTE
- FLOAT4
- FLOAT8
- POUR
- OBLIGER
- ÉTRANGER
- DE
- TEXTE INTÉGRAL
- GÉNÉRÉ

- OBTENIR
- SUBVENTION
- GROUPE
- AYANT
- HAUTE PRIORITÉ
- HOUR\_MICROSECOND
- HOUR\_MINUTE
- HOUR\_SECOND
- SI
- IGNORER
- DANS
- INDICE
- DANS LE FICHER
- INTERNE
- De tout
- INSENSIBLE
- INSÉRER
- INT
- INT1
- INT2
- INT3
- INT4
- INT8
- ENTIER
- INTERVALLE
- DANS
- IO\_AFTER\_GTIDS
- IO\_BEFORE\_GTIDS
- EST
- RÉPÉTER
- JOINDRE
- CLÉ
- CLÉS
- TUER
- DE PREMIER PLAN
- LAISSER
- LA GAUCHE
- COMME
- LIMITE
- LINÉAIRE
- LIGNES
- CHARGE
- HEURE LOCALE
- LOCALTIMESTAMP
- FERMER À CLÉ
- LONGUE

- LONGBLOB
- LONGTEXT
- BOUCLE
- PRIORITÉ BASSE
- MASTER\_BIND
- MASTER\_SSL\_VERIFY\_SERVER\_CERT
- RENCONTRE
- VALEUR MAX
- MEDIUMBLOB
- MEDIUMINT
- MEDIUMTEXT
- MIDDLEINT
- MINUTE\_MICROSECOND
- MINUTE\_SECOND
- MOD
- MODIFIE
- NATUREL
- NE PAS
- NO\_WRITE\_TO\_BINLOG
- NUL
- NUMÉRIQUE
- SUR
- OPTIMISER
- OPTIMIZER\_COSTS
- OPTION
- EN OPTION
- OU
- COMMANDE
- EN DEHORS
- EXTÉRIEUR
- Sortie
- CLOISON
- PRÉCISION
- PRIMAIRE
- PROCÉDURE
- PURGE
- GAMME
- LIS
- LISE
- LIRE ÉCRIRE
- RÉAL
- LES RÉFÉRENCES
- REGEXP
- LIBÉRATION
- RENOMMER
- RÉPÉTER

- REMPLACER
- EXIGER
- RESIGNAL
- RESTREINDRE
- REVENIR
- RÉVOQUER
- DROITE
- RLIKE
- SCHÉMA
- SCHEMAS
- SECOND\_MICROSECOND
- SÉLECTIONNER
- SENSIBLE
- SÉPARATEUR
- ENSEMBLE
- MONTRER
- SIGNAL
- SMALLINT
- SPATIAL
- SPÉCIFIQUE
- SQL
- SQLEXCEPTION
- SQLSTATE
- SQLWARNING
- SQL\_BIG\_RESULT
- SQL\_CALC\_FOUND\_ROWS
- SQL\_SMALL\_RESULT
- SSL
- DÉPART
- STOCKÉ
- STRAIGHT\_JOIN
- TABLE
- TERMINÉ
- PUIS
- TINYBLOB
- TINYINT
- TINYTEXTE
- À
- Trailer
- DÉCLENCHEUR
- VRAI
- ANNULER
- SYNDICAT
- UNIQUE
- OUVRIR
- NON SIGNÉ

- METTRE À JOUR
- USAGE
- UTILISATION
- EN UTILISANT
- UTC\_DATE
- UTC\_TIME
- UTC\_TIMESTAMP
- VALEURS
- VARBINARY
- VARCHAR
- VARCHARACTER
- VARIATION
- VIRTUEL
- QUAND
- OÙ
- TANDIS QUE
- AVEC
- ÉCRIRE
- XOR
- ANNÉE MOIS
- ZEROFILL
- GÉNÉRÉ
- OPTIMIZER\_COSTS
- STOCKÉ
- VIRTUEL

## Exemples

### Erreurs dues aux mots réservés

Lorsque vous essayez de sélectionner dans un tableau appelé `order` comme celui-ci

```
select * from order
```

l'erreur augmente:

Code d'erreur: 1064. Vous avez une erreur dans votre syntaxe SQL; vérifiez le manuel qui correspond à votre version du serveur MySQL pour la bonne syntaxe à utiliser près de 'order' à la ligne 1

Les mots-clés réservés dans MySQL doivent être échappés avec des backticks ( ` ` )

```
select * from `order`
```

faire la distinction entre un mot-clé et un nom de table ou de colonne.

Voir aussi: [Erreur de syntaxe due à l'utilisation d'un mot réservé comme nom de table ou de](#)

colonne dans MySQL .

Lire Mots réservés en ligne: <https://riptutorial.com/fr/mysql/topic/1398/mots-reserves>



---

# Chapitre 39: MySQL Admin

## Exemples

### Changer le mot de passe root

```
mysqladmin -u root -p'old-password' password 'new-password'
```

### Déposer la base de données

Utile pour les scripts permettant de supprimer toutes les tables et de supprimer la base de données:

```
mysqladmin -u[username] -p[password] drop [database]
```

Utilisez avec une extrême prudence.

Pour `DROP` base de données en tant que script SQL (vous aurez besoin du privilège `DROP` sur cette base de données):

```
DROP DATABASE database_name
```

ou

```
DROP SCHEMA database_name
```

### Atomic RENAME & Rechargement de table

```
RENAME TABLE t TO t_old, t_copy TO t;
```

Aucune autre session ne peut accéder aux tables impliquées pendant l'exécution de `RENAME TABLE`. L'opération de renommage n'est donc pas soumise à des problèmes de concurrence.

Atomic Rename est surtout pour recharger complètement une table sans attendre `DELETE` et charger pour terminer:

```
CREATE TABLE new LIKE real;
load `new` by whatever means - LOAD DATA, INSERT, whatever
RENAME TABLE real TO old, new TO real;
DROP TABLE old;
```

Lire MySQL Admin en ligne: <https://riptutorial.com/fr/mysql/topic/2991/mysql-admin>

---

# Chapitre 40: MySQL LOCK TABLE

## Syntaxe

- LOCK TABLES nom\_table [READ | ÉCRIRE]; // Lock Table
- DÉVERROUILLEZ LES TABLES; // Déverrouiller les tables

## Remarques

Le verrouillage est utilisé pour résoudre les problèmes de concurrence. Le verrouillage est requis uniquement lors de l'exécution d'une transaction, qui lit d'abord une valeur à partir d'une base de données et, plus tard, écrit cette valeur dans la base de données. Les verrous ne sont jamais requis pour les opérations d'insertion, de mise à jour ou de suppression autonomes.

Il existe deux types de serrures disponibles

READ LOCK - lorsqu'un utilisateur lit uniquement depuis une table.

WRITE LOCK - lorsqu'un utilisateur lit et écrit à la fois un tableau.

Lorsqu'un utilisateur détient un `WRITE LOCK` sur une table, aucun autre utilisateur ne peut lire ou écrire sur cette table. Lorsqu'un utilisateur détient un `READ LOCK` sur une table, d'autres utilisateurs peuvent également lire ou maintenir un `READ LOCK`, mais aucun utilisateur ne peut écrire ou maintenir un `WRITE LOCK` sur cette table.

Si le moteur de stockage par défaut est InnoDB, MySQL utilise automatiquement le verrouillage au niveau des lignes afin que plusieurs transactions puissent utiliser la même table simultanément pour la lecture et l'écriture, sans se faire mutuellement attendre.

Pour tous les moteurs de stockage autres qu'InnoDB, MySQL utilise le verrouillage de table.

Pour plus de détails sur le verrouillage de la table [Voir ici](#)

## Exemples

### Mysql Serrures

*Les verrous de table peuvent être un outil important pour `ENGINE=MyISAM`, mais sont rarement utiles pour `ENGINE=InnoDB`. Si vous êtes tenté d'utiliser des verrous de table avec InnoDB, vous devez repenser la façon dont vous travaillez avec les transactions.*

MySQL permet aux sessions client d'acquérir explicitement des verrous de table dans le but de coopérer avec d'autres sessions pour accéder aux tables ou d'empêcher d'autres sessions de modifier des tables pendant les périodes où une session nécessite un accès exclusif à ces dernières. Une session peut acquérir ou libérer des verrous uniquement pour elle-même. Une

session ne peut pas acquérir de verrous pour une autre session ou libérer des verrous détenus par une autre session.

Les verrous peuvent être utilisés pour émuler des transactions ou pour accélérer la mise à jour des tableaux. Ceci est expliqué plus en détail plus loin dans cette section.

**Commande:** `LOCK TABLES table_name READ|WRITE;`

vous ne pouvez affecter que le type de verrou à une seule table;

**Exemple (READ LOCK):**

```
LOCK TABLES table_name READ;
```

**Exemple (WRITE LOCK):**

```
LOCK TABLES table_name WRITE;
```

Pour voir que le verrou est appliqué ou non, utilisez la commande suivante

```
SHOW OPEN TABLES;
```

Pour vider / supprimer tous les verrous, utilisez la commande suivante:

```
UNLOCK TABLES;
```

**EXEMPLE:**

```
LOCK TABLES products WRITE;  
INSERT INTO products(id,product_name) SELECT id,old_product_name FROM old_products;  
UNLOCK TABLES;
```

Par exemple, toute connexion externe ne peut écrire aucune donnée dans la table des produits avant de déverrouiller le produit de la table

**EXEMPLE:**

```
LOCK TABLES products READ;  
INSERT INTO products(id,product_name) SELECT id,old_product_name FROM old_products;  
UNLOCK TABLES;
```

Par exemple, toute connexion externe ne peut lire aucune donnée de la table des produits avant de déverrouiller le produit de la table

## Verrouillage au niveau des lignes

Si les tables utilisent InnoDB, MySQL utilise automatiquement le verrouillage au niveau des lignes afin que plusieurs transactions puissent utiliser la même table simultanément pour la lecture et l'écriture, sans se faire mutuellement attendre.

Si deux transactions tentent de modifier la même ligne et que les deux utilisent le verrouillage au niveau des lignes, l'une des transactions attend que l'autre se termine.

Le verrouillage au niveau des lignes peut également être obtenu en utilisant l' `SELECT ... FOR UPDATE` pour chaque ligne à modifier.

Considérons deux connexions pour expliquer le verrouillage au niveau des lignes en détail

### Connexion 1

```
START TRANSACTION;
SELECT ledgerAmount FROM accDetails WHERE id = 1 FOR UPDATE;
```

Dans la connexion 1, le verrouillage de niveau de ligne est obtenu par l' `SELECT ... FOR UPDATE` .

### Connexion 2

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 500 WHERE id=1;
```

Lorsque quelqu'un essaie de mettre à jour la même ligne dans la connexion 2, cela attendra que la connexion 1 termine la transaction ou le message d'erreur sera affiché en fonction du paramètre `innodb_lock_wait_timeout` , qui par défaut est 50 secondes.

```
Error Code: 1205. Lock wait timeout exceeded; try restarting transaction
```

Pour afficher les détails de ce verrou, exécutez `SHOW ENGINE INNODB STATUS`

```
---TRANSACTION 1973004, ACTIVE 7 sec updating
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 360, 1 row lock(s)
MySQL thread id 4, OS thread handle 0x7f996beac700, query id 30 localhost root update
UPDATE accDetails SET ledgerAmount = ledgerAmount + 500 WHERE id=1
----- TRX HAS BEEN WAITING 7 SEC FOR THIS LOCK TO BE GRANTED:
```

### Connexion 2

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 250 WHERE id=2;
1 row(s) affected
```

Mais lors de la mise à jour d'une autre ligne de la connexion 2 sera exécutée sans aucune erreur.

### Connexion 1

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 750 WHERE id=1;
COMMIT;
1 row(s) affected
```

Maintenant, le verrouillage de ligne est libéré, car la transaction est validée dans la connexion 1.

### Connexion 2

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 500 WHERE id=1;
1 row(s) affected
```

La mise à jour est exécutée sans aucune erreur dans la connexion 2 après le verrouillage de la ligne de la connexion 1 en terminant la transaction.

Lire MySQL LOCK TABLE en ligne: <https://riptutorial.com/fr/mysql/topic/5233/mysql-lock-table>

---

# Chapitre 41: Mysql Performance Tips

## Exemples

### Sélectionner l'optimisation de la déclaration

Vous trouverez ci-dessous quelques astuces à retenir lorsque vous écrivez une requête select dans MySQL qui peut nous aider à réduire le temps de requête: -

1. Chaque fois que nous utilisons où dans une grande table, nous devons nous assurer que la colonne dans la clause where est indexée ou non. Ex: - Sélectionnez \* from employee où user\_id > 2000. user\_id si indexé accélère alors l'évaluation de la requête atlot. Les index sont également très importants lors des jointures et des clés étrangères.
2. Lorsque vous avez besoin de la plus petite section de contenu plutôt que d'en extraire des données entières, essayez d'utiliser limit. Plutôt que d'écrire Ex: - Sélectionnez \* de l'employé. Si vous n'avez besoin que de 20 premiers employés de lakhs, utilisez simplement la limite Ex: - Sélectionnez \* de l'employé LIMIT 20.
3. Vous pouvez également optimiser votre requête en fournissant le nom de colonne souhaité dans le jeu de résultats. Plutôt que d'écrire Ex: - Sélectionnez \* de l'employé. Il suffit de mentionner le nom de la colonne à partir de laquelle vous avez besoin de données si votre table contient beaucoup de colonnes et que vous souhaitez en avoir quelques-unes. Ex: - Sélectionnez id, nom de l'employé.
4. Colonne d'index si vous utilisez pour vérifier la valeur NULL dans la clause where. Si vous avez une déclaration en tant que SELECT \* FROM nom\_de\_table WHERE key\_col IS NULL; alors si key\_col est indexé alors la requête sera évaluée plus rapidement.

### Optimisation de la disposition de stockage pour les tables InnoDB

1. Dans InnoDB, avoir une longue PRIMARY KEY (soit une seule colonne avec une longue valeur, soit plusieurs colonnes qui forment une longue valeur composite) gaspille beaucoup d'espace disque. La valeur de clé primaire pour une ligne est dupliquée dans tous les enregistrements d'index secondaires qui pointent vers la même ligne. Créez une colonne AUTO\_INCREMENT comme clé primaire si votre clé primaire est longue.
2. Utilisez le type de données VARCHAR au lieu de CHAR pour stocker des chaînes de longueur variable ou des colonnes avec plusieurs valeurs NULL. Une colonne CHAR (N) prend toujours N caractères pour stocker des données, même si la chaîne est plus courte ou sa valeur est NULL. Les tables plus petites s'intègrent mieux dans le pool de mémoire tampon et réduisent les E / S disque.

Lorsque vous utilisez un format de ligne COMPACT (format InnoDB par défaut) et des jeux de caractères de longueur variable, tels que utf8 ou sjis, les colonnes CHAR (N) occupent un espace variable, mais contiennent au moins N octets.

3. Pour les tables volumineuses ou contenant beaucoup de données textuelles ou numériques répétitives, envisagez d'utiliser le format de ligne COMPRESSED. Moins d'E / S de disque sont nécessaires pour importer des données dans le pool de mémoire tampon ou pour effectuer des analyses de table complètes. Avant de prendre une décision définitive, mesurez la quantité de compression que vous pouvez obtenir en utilisant le format de ligne COMPRESSED versus COMPACT. *Avertissement:* Les tests de performances montrent rarement une compression supérieure à 2: 1 et il y a beaucoup de surcharge dans `buffer_pool` pour COMPRESSED.
4. Une fois que vos données atteignent une taille stable ou qu'une table croissante a augmenté de dizaines ou de centaines de mégaoctets, envisagez d'utiliser la commande OPTIMIZE TABLE pour réorganiser la table et compacter tout espace perdu. Les tables réorganisées nécessitent moins d'E / S disque pour effectuer des analyses de table complètes. Il s'agit d'une technique simple qui peut améliorer les performances lorsque d'autres techniques, telles que l'amélioration de l'utilisation des index ou le réglage du code d'application, ne sont pas pratiques. *Avertissement :* Indépendamment de la taille de la table, OPTIMIZE TABLE doit être rarement exécuté. C'est parce que c'est coûteux et améliore rarement la table suffisamment pour en valoir la peine. InnoDB est raisonnablement bon pour garder ses arbres B + exempts de beaucoup d'espace perdu.

OPTIMIZE TABLE copie la partie données de la table et reconstruit les index. Les avantages proviennent de l'amélioration de l'emballage des données au sein des index et de la fragmentation réduite au sein des espaces de table et sur le disque. Les avantages varient en fonction des données de chaque tableau. Vous pouvez constater qu'il y a des gains significatifs pour certains et pas pour d'autres, ou que les gains diminuent avec le temps jusqu'à ce que vous optimisiez ensuite le tableau. Cette opération peut être lente si la table est volumineuse ou si les index en cours de reconstruction ne rentrent pas dans le pool de mémoire tampon. La première exécution après avoir ajouté beaucoup de données à une table est souvent beaucoup plus lente que les exécutions ultérieures.

## Construire un index composite

Dans de nombreuses situations, un index composite est plus performant qu'un index avec une seule colonne. Pour créer un index composite optimal, remplissez-le avec des colonnes dans cet ordre.

- = colonne (s) de la clause WHERE premier. (par exemple, INDEX(a,b,...) pour WHERE a=12 AND b='xyz' ... )
- Colonne (s) IN ; l'optimiseur peut être en mesure de sauter l'index.
- Une "plage" (par exemple x BETWEEN 3 AND 9 , name LIKE 'J%' ) Elle n'utilisera rien après la première colonne de la plage.
- Toutes les colonnes de GROUP BY , dans l'ordre
- Toutes les colonnes dans ORDER BY , dans l'ordre. Fonctionne uniquement si tous sont en ASC ou si tous sont DESC ou si vous utilisez 8.0.

Notes et exceptions:

- Ne dupliquez aucune colonne.
- Ignorez tous les cas qui ne s'appliquent pas.
- Si vous n'utilisez pas toutes les colonnes de `WHERE` , il n'est pas nécessaire de passer à `GROUP BY` , etc.
- Il y a des cas où il est utile d'indexer uniquement la ou les colonnes `ORDER BY` , en ignorant `WHERE` .
- Ne "cachez" pas une colonne dans une fonction (par exemple, `DATE(x) = ...` ne peut pas utiliser `x` dans l'index).
- L'indexation du préfixe (par exemple, `text_col(99)` ) ne sera probablement pas utile. peut faire mal

*Plus de détails et de conseils .*

Lire Mysql Performance Tips en ligne: <https://riptutorial.com/fr/mysql/topic/5752/mysql-performance-tips>



# Chapitre 42: mysqlimport

## Paramètres

| Paramètre                                    | La description  |
|--|---|
| <code>--delete -D</code>                     | vider la table avant d'importer le fichier texte            |
| <code>--fields-optionally-enclosed-by</code> | définir le caractère qui cite les champs                    |
| <code>--fields-terminated-by</code>          | terminateur de champ  |
| <code>--ignore -i</code>                     | ignorer la ligne ingérée en cas de doublons                 |
| <code>--lines-terminated-by</code>           | définir le terminateur de ligne                             |
| <code>--password -p</code>                   | mot de passe  |
| <code>--port -P</code>                       | Port  |
| <code>--replace -r</code>                    | écraser l'ancienne ligne d'entrée en cas de clés dupliquées |
| <code>--user -u</code>                       | Nom d'utilisateur   |
| <code>--where -w</code>                      | spécifier une condition                                     |

## Remarques

`mysqlimport` utilisera le nom du fichier importé, après la suppression de l'extension, pour déterminer la table de destination.

## Exemples

### Utilisation de base

Compte tenu du fichier séparé par tabs `employee.txt`

- `\t Arthur Dent`
- `\t Marvin`
- `\t Zaphod Beeblebrox`

```
$ mysql --user=user --password=password mycompany -e 'CREATE TABLE employee(id INT, name VARCHAR(100), PRIMARY KEY (id))'
```

```
$ mysqlimport --user=user --password=password mycompany employee.txt
```

## Utiliser un délimiteur de champ personnalisé

Étant donné le fichier texte `employee.txt`

```
1 | Arthur Dent
2 | Marvin
3 | Zaphod Beeblebrox
```

```
$ mysqlimport --fields-terminated-by='|' mycompany employee.txt
```

## Utiliser un délimiteur de ligne personnalisé

Cet exemple est utile pour les terminaisons de type Windows:

```
$ mysqlimport --lines-terminated-by='\r\n' mycompany employee.txt
```

## Gestion des clés en double

Donné la table `Employee`

| id | prénom       |
|----|--------------|
| 3  | Yooden Vranx |

Et le fichier `employee.txt`

```
1 \t Arthur Dent
2 \t Marvin
3 \t Zaphod Beeblebrox
```

L'option `--ignore` ignorera l'entrée sur les clés en double

```
$ mysqlimport --ignore mycompany employee.txt
```

| id | prénom       |
|----|--------------|
| 1  | Arthur Dent  |
| 2  | Marvin       |
| 3  | Yooden Vranx |

L'option `--replace` remplacera l'ancienne entrée

```
$ mysqlimport --replace mycompany employee.txt
```

| id | prénom            |
|----|-------------------|
| 1  | Arthur Dent       |
| 2  | Marvin            |
| 3  | Zaphod Beeblebrox |

## Import conditionnel

```
$ mysqlimport --where="id>2" mycompany employee.txt
```

## Importer un csv standard

```
$ mysqlimport  
  --fields-optionally-enclosed-by='''  
  --fields-terminated-by=,  
  --lines-terminated-by="\r\n"  
mycompany employee.csv
```

Lire mysqlimport en ligne: <https://riptutorial.com/fr/mysql/topic/5215/mysqlimport>

---

# Chapitre 43: NUL

## Exemples

### Utilise NULL

- Données pas encore connues - telles que `end_date` , `rating`
- Données facultatives - telles que `middle_initial` (bien que cela puisse être mieux que la chaîne vide)
- 0/0 - Résultat de certains calculs, tels que zéro divisé par zéro.
- NULL n'est pas égal à "" (chaîne vide) ou 0 (en cas de nombre entier).
- autres?

### Test NULL

- `IS NULL / IS NOT NULL - = NULL` ne fonctionne pas comme prévu.
- `x <=> y` est une comparaison "null-safe".

Dans un test `LEFT JOIN` pour les lignes d' `a` pour lesquelles il n'y a *pas de* ligne correspondante dans `b` .

```
SELECT ...
  FROM a
  LEFT JOIN b ON ...
 WHERE b.id IS NULL
```

Lire NUL en ligne: <https://riptutorial.com/fr/mysql/topic/6757/nul>

# Chapitre 44: Opérations de chaîne

## Paramètres

| prénom              | La description   |
|---------------------|--|
| ASCII ()            | Renvoie la valeur numérique du caractère le plus à gauche  |
| POUBELLE()          | Renvoie une chaîne contenant une représentation binaire d'un nombre  |
| BIT_LENGTH ()       | Renvoie la longueur de l'argument en bits  |
| CARBONISER()        | Renvoie le caractère pour chaque entier passé  |
| CHAR_LENGTH ()      | Renvoyer le nombre de caractères dans l'argument   |
| CHARACTER_LENGTH () | Synonyme de CHAR_LENGTH ()   |
| CONCAT ()           | Retourne la chaîne concaténée  |
| CONCAT_WS ()        | Retour concaténer avec séparateur  |
| ELT ()              | Renvoyer la chaîne au numéro d'index   |
| EXPORT_SET ()       | Renvoie une chaîne de telle sorte que pour chaque bit défini dans les bits de valeur, vous obtenez une chaîne de caractères et pour chaque bit non défini, vous obtenez une chaîne désactivée. |
| CHAMP()             | Renvoie l'index (position) du premier argument dans les arguments suivants   |
| FIND_IN_SET ()      | Renvoie la position d'index du premier argument dans le second argument  |
| FORMAT()            | Retourne un nombre mis en forme au nombre spécifié de décimales  |
| FROM_BASE64 ()      | Décoder en une chaîne de base 64 et retourner le résultat  |
| HEX ()              | Renvoie une représentation hexadécimale d'une valeur décimale ou de chaîne   |
| INSÉRER()           | Insérer une sous-chaîne à la position spécifiée jusqu'au nombre de caractères spécifié   |

| prénom                 | La description  |
|------------------------|---|
| INSTR ()               | Renvoie l'index de la première occurrence de la sous-chaîne                                     |
| LCASE ()               | Synonyme de LOWER ()  |
| LA GAUCHE()            | Renvoie le nombre de caractères le plus à gauche spécifié                                       |
| LONGUEUR()             | Renvoie la longueur d'une chaîne en octets  |
| COMME                  | Correspondance simple   |
| FICHER DE CHARGEMENT() | Charger le fichier nommé  |
| LOCALISER()            | Renvoie la position de la première occurrence de la sous-chaîne                                 |
| INFÉRIEUR()            | Renvoie l'argument en minuscule   |
| LPAD ()                | Renvoie l'argument de chaîne, rempli à gauche avec la chaîne spécifiée                          |
| LTRIM ()               | Supprimer les espaces de début  |
| MAKE_SET ()            | Retourne un ensemble de chaînes séparées par des virgules dont le bit correspondant est en bits |
| RENCONTRE              | Effectuer une recherche en texte intégral   |
| MILIEU()               | Renvoie une sous-chaîne à partir de la position spécifiée                                       |
| PAS COMME              | Négation de correspondance de modèle simple   |
| PAS REGEXP             | Négation de REGEXP  |
| OCT()                  | Renvoie une chaîne contenant une représentation octale d'un nombre                              |
| OCTET_LENGTH ()        | Synonyme de LENGTH ()   |
| ORD ()                 | Renvoie le code de caractère pour le caractère le plus à gauche de l'argument                   |
| POSITION()             | Synonyme de LOCATE ()   |
| CITATION()             | Échapper l'argument pour l'utiliser dans une instruction SQL                                    |
| REGEXP                 | Correspondance de motif à l'aide d'expressions régulières                                       |
| RÉPÉTER()              | Répéter une chaîne le nombre de fois spécifié   |
| REPLACER()             | Remplace les occurrences d'une chaîne spécifiée   |

| prénom             | La description  |
|--------------------|---|
| SENS INVERSE()     | Inverser les caractères dans une chaîne   |
| DROITE()           | Renvoie le nombre de caractères le plus à droite spécifié                                 |
| RLIKE              | Synonyme de REGEXP  |
| RPAD ()            | Ajouter une chaîne au nombre de fois spécifié   |
| RTRIM ()           | Supprimer les espaces de fin  |
| SOUNDEX ()         | Retourne une chaîne soundex   |
| SONS COMME         | Comparer les sons   |
| ESPACE()           | Renvoie une chaîne du nombre d'espaces spécifié   |
| STRCMP ()          | Comparer deux chaînes   |
| SUBSTR ()          | Renvoie la sous-chaîne comme spécifié   |
| SUBSTRING ()       | Renvoie la sous-chaîne comme spécifié   |
| SUBSTRING_INDEX () | Renvoie une sous-chaîne d'une chaîne avant le nombre d'occurrences spécifié du délimiteur |
| TO_BASE64 ()       | Renvoie l'argument converti en chaîne de base 64  |
| RÉDUIRE()          | Supprimer les espaces de début et de fin  |
| UCASE ()           | Synonyme de UPPER ()  |
| UNHEX ()           | Renvoie une chaîne contenant une représentation hexadécimale d'un nombre                  |
| PLUS HAUT()        | Convertir en majuscule  |
| WEIGHT_STRING ()   | Renvoie la chaîne de poids pour une chaîne  |

## Exemples

### Rechercher un élément dans une liste séparée par des virgules

```
SELECT FIND_IN_SET('b', 'a,b,c');
```

Valeur de retour:

2

```
SELECT FIND_IN_SET('d', 'a,b,c');
```

Valeur de retour:

0

## STR\_TO\_DATE - Convertit la chaîne à la date

Avec une colonne de l'un des types de chaîne, nommée `my_date_field` avec une valeur telle que [la chaîne] `07/25/2016`, l'instruction suivante illustre l'utilisation de la fonction `STR_TO_DATE` :

```
SELECT STR_TO_DATE(my_date_field, '%m/%d/%Y') FROM my_table;
```

Vous pouvez également utiliser cette fonction dans la clause `WHERE` .

## LOWER () / LCASE ()

Convertir en minuscule l'argument de chaîne

Syntaxe: `LOWER (str)`

```
LOWER('fOoBar') -- 'foobar'
LCASE('fOoBar') -- 'foobar'
```

## REPLACER()

Convertir en minuscule l'argument de chaîne

Syntaxe: `REPLACE (str, from_str, to_str)`

```
REPLACE('foobarbaz', 'bar', 'BAR') -- 'fooBARbaz'
REPLACE('foobarbaz', 'zzz', 'ZZZ') -- 'foobarbaz'
```

## SUBSTRING ()

`SUBSTRING` (ou équivalent: `SUBSTR`) renvoie la sous-chaîne à partir de la position spécifiée et, éventuellement, à la longueur spécifiée

Syntaxe: `SUBSTRING(str, start_position)`

```
SELECT SUBSTRING('foobarbaz', 4); -- 'barbaz'
SELECT SUBSTRING('foobarbaz' FROM 4); -- 'barbaz'

-- using negative indexing
SELECT SUBSTRING('foobarbaz', -6); -- 'barbaz'
SELECT SUBSTRING('foobarbaz' FROM -6); -- 'barbaz'
```

Syntaxe: `SUBSTRING(str, start_position, length)`



```
SELECT SUBSTRING('foobarbaz', 4, 3); -- 'bar'
SELECT SUBSTRING('foobarbaz', FROM 4 FOR 3); -- 'bar'

-- using negative indexing
SELECT SUBSTRING('foobarbaz', -6, 3); -- 'bar'
SELECT SUBSTRING('foobarbaz' FROM -6 FOR 3); -- 'bar'
```

## UPPER () / UCASE ()

Convertir en majuscule l'argument de chaîne

Syntaxe: UPPER (str)

```
UPPER('fOoBar') -- 'FOOBAR'
UCASE('fOoBar') -- 'FOOBAR'
```

## LONGUEUR()

Renvoie la longueur de la chaîne en octets. Étant donné que certains caractères peuvent être encodés en utilisant plus d'un octet, si vous voulez la longueur en caractères, voyez `CHAR_LENGTH ()`

Syntaxe: LONGUEUR (str)

```
LENGTH('foobar') -- 6
LENGTH('fööbar') -- 8 -- contrast with CHAR_LENGTH(...) = 6
```

## CHAR\_LENGTH ()

Renvoie le nombre de caractères de la chaîne

Syntaxe: CHAR\_LENGTH (str)

```
CHAR_LENGTH('foobar') -- 6
CHAR_LENGTH('fööbar') -- 6 -- contrast with LENGTH(...) = 8
```

## HEX (str)

Convertissez l'argument en hexadécimal. Ceci est utilisé pour les chaînes.

```
HEX('fööbar') -- 66F6F6626172 -- in "CHARACTER SET latin1" because "F6" is hex for ö
HEX('fööbar') -- 66C3B6C3B6626172 -- in "CHARACTER SET utf8 or utf8mb4" because "C3B6" is hex for ö
```

Lire Opérations de chaîne en ligne: <https://riptutorial.com/fr/mysql/topic/1399/operations-de-chaine>

# Chapitre 45: Opérations de date et heure

## Exemples

### À présent()

```
Select Now();
```

Affiche la date et l'heure actuelles du serveur.

```
Update `footable` set mydatefield = Now();
```

Cela mettra à jour le champ `mydatefield` avec la date et l'heure du serveur en cours dans le fuseau horaire configuré du serveur, par exemple

```
'2016-07-21 12:00:00'
```

### Arithmétique de date

```
NOW() + INTERVAL 1 DAY -- This time tomorrow
```

```
CURDATE() - INTERVAL 4 DAY -- Midnight 4 mornings ago
```

Affichez les questions mysql stockées qui ont été posées il y a 3 à 10 heures (il y a 180 à 600 minutes):

```
SELECT qId,askDate,minuteDiff
FROM
(
  SELECT qId,askDate,
    TIMESTAMPDIFF(MINUTE,askDate,now()) as minuteDiff
  FROM questions_mysql
) xDerived
WHERE minuteDiff BETWEEN 180 AND 600
ORDER BY qId DESC
LIMIT 50;
```

```
+-----+-----+-----+
| qId      | askDate                | minuteDiff |
+-----+-----+-----+
| 38546828 | 2016-07-23 22:06:50 |      182 |
| 38546733 | 2016-07-23 21:53:26 |      195 |
| 38546707 | 2016-07-23 21:48:46 |      200 |
| 38546687 | 2016-07-23 21:45:26 |      203 |
| ...      | | |
+-----+-----+-----+
```

Pages de manuel MySQL pour [TIMESTAMPDIFF\(\)](#) .

**Attention** N'essayez pas d'utiliser des expressions comme `CURDATE() + 1` pour l'arithmétique des

dates dans MySQL. Ils ne renvoient pas ce que vous attendez, surtout si vous êtes habitué au produit de base de données Oracle. Utilisez `CURDATE() + INTERVAL 1 DAY` place.

## Test par rapport à une plage de dates

Bien que cela soit très tentant d'utiliser `BETWEEN ... AND ...` pour une plage de dates, cela pose problème. Au lieu de cela, ce modèle évite la plupart des problèmes:

```
WHERE x >= '2016-02-25'  
AND x < '2016-02-25' + INTERVAL 5 DAY
```

Avantages:

- `BETWEEN` est inclusif, y compris la date finale ou la seconde.
- `23:59:59` est maladroit et faux si vous avez une résolution en microseconde sur un `DATETIME`.
- Ce modèle évite de traiter des années bissextiles et d'autres calculs de données.
- Cela fonctionne si `x` est `DATE`, `DATETIME` ou `TIMESTAMP`.

## SYSDATE (), NOW (), CURDATE ()

```
SELECT SYSDATE ();
```

Cette fonction renvoie la date et l'heure actuelles sous la forme `'YYYY-MM-DD HH:MM:SS'` ou au format `YYYYMMDDHHMMSS`, selon que la fonction est utilisée dans un contexte de chaîne ou numérique. Il renvoie la date et l'heure dans le fuseau horaire actuel.

```
SELECT NOW ();
```

Cette fonction est synonyme de `SYSDATE ()`.

```
SELECT CURDATE ();
```

Cette fonction renvoie la date actuelle, sans aucune heure, sous forme de valeur au `'YYYY-MM-DD'` ou `YYYYMMDD`, selon que la fonction est utilisée dans un contexte de chaîne ou numérique. Il renvoie la date dans le fuseau horaire actuel.

## Extraire la date à partir de la date donnée ou de la date-heure

```
SELECT DATE('2003-12-31 01:02:03');
```

Le résultat sera:

```
2003-12-31
```

## Utilisation d'un index pour une recherche par date et heure

De nombreuses tables de base de données réelles ont de nombreuses lignes avec des valeurs de colonne `DATE` OU `TIMESTAMP` couvrant beaucoup de temps, y compris des années, voire des décennies. Il est souvent nécessaire d'utiliser une clause `WHERE` pour récupérer un sous-ensemble de cette période. Par exemple, nous pourrions vouloir récupérer des lignes pour la date du 1er septembre 2016 à partir d'une table.

Une façon inefficace de le faire est la suivante:

```
WHERE DATE(x) = '2016-09-01' /* slow! */
```

C'est inefficace car il applique une fonction - `DATE()` - aux valeurs d'une colonne. Cela signifie que MySQL doit examiner chaque valeur de `x` et qu'un index ne peut pas être utilisé.

Une meilleure façon de faire l'opération est la suivante:

```
WHERE x >= '2016-09-01'
AND x < '2016-09-01' + INTERVAL 1 DAY
```

Ceci sélectionne une plage de valeurs de `x` située n'importe où le jour en question, jusqu'à mais n'incluant pas (donc `<`) minuit le jour suivant.

Si la table possède un index sur la colonne `x`, le serveur de base de données peut effectuer une analyse de plage sur l'index. Cela signifie qu'il peut rapidement trouver la première valeur pertinente de `x`, puis balayer l'index séquentiellement jusqu'à ce qu'il trouve la dernière valeur pertinente. Une analyse de plage d'index est beaucoup plus efficace que l'analyse de table complète requise par `DATE(x) = '2016-09-01'`.

Ne soyez pas tenté de l'utiliser, même si cela semble plus efficace.

```
WHERE x BETWEEN '2016-09-01' AND '2016-09-01' + INTERVAL 1 DAY /* wrong! */
```

Il a la même efficacité que l'analyse de distance, mais il sélectionnera les lignes avec des valeurs de `x` tombant exactement à minuit le 2-sept-2016, ce qui n'est pas ce que vous voulez.

Lire Opérations de date et heure en ligne: <https://riptutorial.com/fr/mysql/topic/1882/operations-de-date-et-heure>

# Chapitre 46: Par groupe

## Syntaxe

1. SELECT expression1, expression2, ... expression\_n,
2. agrégat\_fonction (expression)
3. Des tables
4. [O conditions conditions]
5. GROUP BY expression1, expression2, ... expression\_n;

## Paramètres

| Paramètre  | DÉTAILS  |
|--|--|
| expression1,<br>expression2, ...<br>expression_n | Les expressions qui ne sont pas encapsulées dans une fonction d'agrégat doivent être incluses dans la clause GROUP BY.             |
| l'agrégat  | Une fonction telle que SUM, COUNT, MIN, MAX ou AVG.  |
| les tables                                       | Les tableaux que vous souhaitez récupérer des enregistrements. Il doit y avoir au moins une table répertoriée dans la clause FROM. |
| O conditions conditions                          | Optionnel. Les conditions à remplir pour que les enregistrements soient sélectionnés.  |

## Remarques

La clause MySQL GROUP BY est utilisée dans une instruction SELECT pour collecter des données sur plusieurs enregistrements et regrouper les résultats par une ou plusieurs colonnes.

Son comportement est en partie régi par la valeur de [la variable ONLY\\_FULL\\_GROUP\\_BY](#) . Lorsque cette option est activée, les `SELECT` regroupées par colonne ne figurant pas dans la sortie renvoient une erreur. ( [Il s'agit de la valeur par défaut à partir du 5.7.5](#) .) Le fait de définir et de ne pas définir cette variable peut entraîner des problèmes pour les utilisateurs naïfs ou les utilisateurs habitués à d'autres SGBD.

## Exemples

### GROUPE EN UTILISANT LA Fonction SUM

```
SELECT product, SUM(quantity) AS "Total quantity"
```

```
FROM order_details
GROUP BY product;
```

## Groupe en utilisant la fonction MIN

Supposons une table d'employés dans laquelle chaque ligne est un employé qui a un `name`, un `department` et un `salary`.

```
SELECT department, MIN(salary) AS "Lowest salary"
FROM employees
GROUP BY department;
```

Cela vous dira quel service contient l'employé avec le salaire le plus bas et quel est ce salaire. Trouver le `name` de l'employé avec le salaire le plus bas dans chaque département est un problème différent, au-delà de la portée de cet exemple. Voir "groupwise max".

## GRUPE EN UTILISANT LA FONCTION COUNT

```
SELECT department, COUNT(*) AS "Man_Power"
FROM employees
GROUP BY department;
```

## GROUP BY en utilisant HAVING

```
SELECT department, COUNT(*) AS "Man_Power"
FROM employees
GROUP BY department
HAVING COUNT(*) >= 10;
```

L'utilisation de `GROUP BY ... HAVING` pour filtrer les enregistrements agrégés est analogue à l'utilisation de `SELECT ... WHERE` pour filtrer des enregistrements individuels.

Vous pouvez aussi dire `HAVING Man_Power >= 10` car `HAVING` comprend les "alias".

## Grouper en utilisant Group Concat

**Group Concat** est utilisé dans MySQL pour obtenir des valeurs concaténées d'expressions comportant plus d'un résultat par colonne. Signification, il y a beaucoup de lignes à sélectionner pour une colonne telle que `Name(1):Score(*)`

| prénom | But  |
|--------|------|
| Adam   | Un + |
| Adam   | UNE- |
| Adam   | B    |
| Adam   | C +  |

| prénom  | But  |
|---------|------|
| Facture | RÉ-  |
| John    | UNE- |

```
SELECT Name, GROUP_CONCAT(Score ORDER BY Score desc SEPERATOR ' ') AS Grades
FROM Grade
GROUP BY Name
```

### Résultats:

```
+-----+-----+
| Name | Grades |
+-----+-----+
| Adam | C+ B A- A+ |
| Bill | D-      |
| John | A-      |
+-----+-----+
```

## GROUP BY avec les fonctions AGGREGATE

### Table COMMANDES

```
+-----+-----+-----+-----+-----+
| orderid | customerid | customer | total | items |
+-----+-----+-----+-----+-----+
| 1 | 1 | Bob | 1300 | 10 |
| 2 | 3 | Fred | 500 | 2 |
| 3 | 5 | Tess | 2500 | 8 |
| 4 | 1 | Bob | 300 | 6 |
| 5 | 2 | Carly | 800 | 3 |
| 6 | 2 | Carly | 1000 | 12 |
| 7 | 3 | Fred | 100 | 1 |
| 8 | 5 | Tess | 11500 | 50 |
| 9 | 4 | Jenny | 200 | 2 |
| 10 | 1 | Bob | 500 | 15 |
+-----+-----+-----+-----+-----+
```

- **COMPTER**

Renvoie le **nombre de lignes** satisfaisant un critère spécifique dans la clause `WHERE` .

Par exemple: nombre de commandes pour chaque client.

```
SELECT customer, COUNT(*) as orders
FROM orders
GROUP BY customer
ORDER BY customer
```

### Résultat:

```
+-----+-----+
```

| customer | orders |
|----------|--------|
| Bob      | 3      |
| Carly    | 2      |
| Fred     | 2      |
| Jenny    | 1      |
| Tess     | 2      |

- **SOMME**

Renvoie la **somme** de la colonne sélectionnée.

Par exemple: somme du total et des articles pour chaque client.

```
SELECT customer, SUM(total) as sum_total, SUM(items) as sum_items
FROM orders
GROUP BY customer
ORDER BY customer
```

**Résultat:**

| customer | sum_total | sum_items |
|----------|-----------|-----------|
| Bob      | 2100      | 31        |
| Carly    | 1800      | 15        |
| Fred     | 600       | 3         |
| Jenny    | 200       | 2         |
| Tess     | 14000     | 58        |

- **AVG**

Renvoie la valeur **moyenne** d'une colonne de valeur numérique.

Par exemple: valeur de commande moyenne pour chaque client.

```
SELECT customer, AVG(total) as avg_total
FROM orders
GROUP BY customer
ORDER BY customer
```

**Résultat:**

| customer | avg_total |
|----------|-----------|
| Bob      | 700       |
| Carly    | 900       |
| Fred     | 300       |
| Jenny    | 200       |
| Tess     | 7000      |



- **MAX**

Renvoie la valeur la **plus** élevée d'une certaine colonne ou expression.

Par exemple: le plus haut total de commande pour chaque client.

```
SELECT customer, MAX(total) as max_total
FROM orders
GROUP BY customer
ORDER BY customer
```

**Résultat:**

```
+-----+-----+
| customer | max_total |
+-----+-----+
| Bob      |      1300 |
| Carly    |      1000 |
| Fred     |       500 |
| Jenny    |       200 |
| Tess     |     11500 |
+-----+-----+
```

- **MIN**

Renvoie la valeur la **plus** basse d'une certaine colonne ou expression.

Par exemple: le plus bas total de commande pour chaque client.

```
SELECT customer, MIN(total) as min_total
FROM orders
GROUP BY customer
ORDER BY customer
```

**Résultat:**

```
+-----+-----+
| customer | min_total |
+-----+-----+
| Bob      |       300 |
| Carly    |       800 |
| Fred     |       100 |
| Jenny    |       200 |
| Tess     |     2500 |
+-----+-----+
```

Lire Par groupe en ligne: <https://riptutorial.com/fr/mysql/topic/3523/par-groupe>

# Chapitre 47: Partitionnement

## Remarques

- **Partitionnement RANGE** Ce type de partitionnement affecte des lignes aux partitions en fonction des valeurs de colonne comprises dans une plage donnée.
- **LIST partitionnement** . Semblable au partitionnement par RANGE, sauf que la partition est sélectionnée en fonction de colonnes correspondant à un ensemble de valeurs discrètes.
- **Partitionnement HASH** . Avec ce type de partitionnement, une partition est sélectionnée en fonction de la valeur renvoyée par une expression définie par l'utilisateur qui opère sur les valeurs de colonne dans les lignes à insérer dans la table. La fonction peut être constituée de toute expression valide dans MySQL qui donne une valeur entière non négative. Une extension de ce type, `LINEAR HASH` , est également disponible.
- **Partitionnement KEY** . Ce type de partitionnement est similaire au partitionnement par HASH, sauf que seules une ou plusieurs colonnes à évaluer sont fournies et que le serveur MySQL fournit sa propre fonction de hachage. Ces colonnes peuvent contenir des valeurs autres que des valeurs entières, car la fonction de hachage fournie par MySQL garantit un résultat entier quel que soit le type de données de la colonne. Une extension à ce type, `LINEAR KEY` , est également disponible.

## Exemples

### Partitionnement RANGE

Une table partitionnée par plage est partitionnée de telle sorte que chaque partition contient des lignes pour lesquelles la valeur de l'expression de partitionnement se situe dans une plage donnée. Les plages doivent être contiguës mais ne se chevauchent pas et sont définies à l'aide de l'opérateur `VALUES LESS THAN` . Pour les exemples suivants, supposez que vous créez un tableau tel que celui-ci pour contenir les dossiers du personnel pour une chaîne de 20 magasins de vidéos, numérotés de 1 à 20:

```
CREATE TABLE employees (  
  id INT NOT NULL,  
  fname VARCHAR(30),  
  lname VARCHAR(30),  
  hired DATE NOT NULL DEFAULT '1970-01-01',  
  separated DATE NOT NULL DEFAULT '9999-12-31',  
  job_code INT NOT NULL,  
  store_id INT NOT NULL  
);
```

Ce tableau peut être partitionné par plage de différentes manières, en fonction de vos besoins. Une façon serait d'utiliser la colonne `store_id` . Par exemple, vous pouvez décider de partitionner la table en ajoutant une clause `PARTITION BY RANGE` , comme indiqué ici:

```
ALTER TABLE employees PARTITION BY RANGE (store_id) (
  PARTITION p0 VALUES LESS THAN (6),
  PARTITION p1 VALUES LESS THAN (11),
  PARTITION p2 VALUES LESS THAN (16),
  PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

`MAXVALUE` représente une valeur entière qui est toujours supérieure à la plus grande valeur entière possible (en langage mathématique, elle sert de limite inférieure).

basé sur [le document officiel MySQL](#) .

## Partitionnement LIST

Le partitionnement de liste est similaire au partitionnement à plusieurs niveaux. Comme dans le partitionnement par `RANGE`, chaque partition doit être explicitement définie. La principale différence entre les deux types de partitionnement réside dans le fait que, dans le partitionnement de liste, chaque partition est définie et sélectionnée en fonction de l'appartenance d'une valeur de colonne dans un ensemble de listes de valeurs plutôt que dans un ensemble de plages contiguës. Ceci est fait en utilisant `PARTITION BY LIST(expr)` où `expr` est une valeur de colonne ou une expression basée sur une valeur de colonne et renvoyant une valeur entière, puis définissant chaque partition au moyen d'une `VALUES IN (value_list)`, où `value_list` est liste d'entiers séparés par des virgules.

Pour les exemples suivants, nous supposons que la définition de base de la table à partitionner est fournie par l'instruction `CREATE TABLE` présentée ici:

```
CREATE TABLE employees (
  id INT NOT NULL,
  fname VARCHAR(30),
  lname VARCHAR(30),
  hired DATE NOT NULL DEFAULT '1970-01-01',
  separated DATE NOT NULL DEFAULT '9999-12-31',
  job_code INT,
  store_id INT
);
```

Supposons qu'il y ait 20 magasins de vidéos répartis sur 4 franchises, comme le montre le tableau suivant.

| Région  | Numéros d'identification de magasin |
|---------|-------------------------------------|
| Nord    | 3, 5, 6, 9, 17                      |
| est     | 1, 2, 10, 11, 19, 20                |
| Ouest   | 4, 12, 13, 14, 18                   |
| Central | 7, 8, 15, 16                        |

Pour partitionner cette table de manière à ce que les lignes des magasins appartenant à la même région soient stockées dans la même partition

```
ALTER TABLE employees PARTITION BY LIST(store_id) (  
    PARTITION pNorth VALUES IN (3,5,6,9,17),  
    PARTITION pEast VALUES IN (1,2,10,11,19,20),  
    PARTITION pWest VALUES IN (4,12,13,14,18),  
    PARTITION pCentral VALUES IN (7,8,15,16)  
);
```

basé sur [le document officiel MySQL](#) .

## Partitionnement HASH

Le partitionnement par HASH est principalement utilisé pour assurer une distribution uniforme des données parmi un nombre prédéterminé de partitions. Avec le partitionnement par plage ou par liste, vous devez spécifier explicitement dans quelle partition une valeur de colonne donnée ou un ensemble de valeurs de colonne doit être stocké; Avec le partitionnement par hachage, MySQL s'en charge pour vous et vous devez uniquement spécifier une valeur ou une expression de colonne basée sur une valeur de colonne à hacher et le nombre de partitions dans lesquelles la table partitionnée doit être divisée.

L'instruction suivante crée une table qui utilise le hachage sur la colonne `store_id` et est divisée en 4 partitions:

```
CREATE TABLE employees (  
    id INT NOT NULL,  
    fname VARCHAR(30),  
    lname VARCHAR(30),  
    hired DATE NOT NULL DEFAULT '1970-01-01',  
    separated DATE NOT NULL DEFAULT '9999-12-31',  
    job_code INT,  
    store_id INT  
)  
PARTITION BY HASH(store_id)  
PARTITIONS 4;
```

Si vous n'incluez pas de clause `PARTITIONS` , le nombre de partitions par défaut est 1.

basé sur [le document officiel MySQL](#) .

Lire Partitionnement en ligne: <https://riptutorial.com/fr/mysql/topic/5128/partitionnement>

# Chapitre 48: Personnaliser PS1

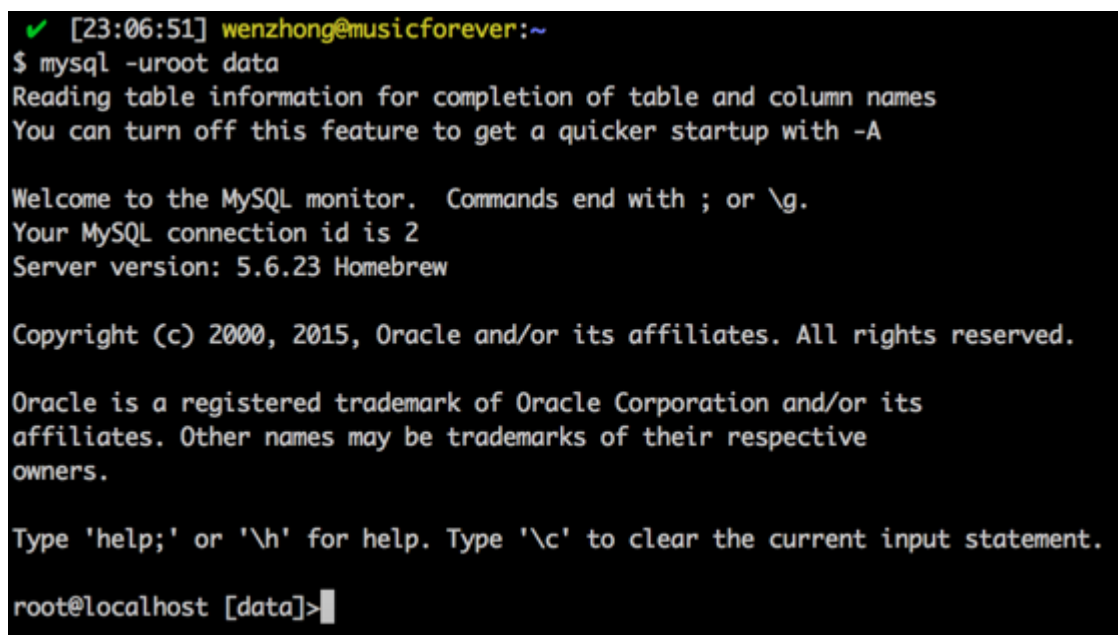
## Exemples

### Personnaliser MySQL PS1 avec la base de données actuelle

Dans le fichier `.bashrc` ou `.bash_profile`, en ajoutant:

```
export MYSQL_PS1="\u@\h [\d]>"
```

faire en sorte que le client MySQL PROMPT affiche l'utilisateur actuel @ host [base de données].



```
✓ [23:06:51] wenzhong@musicforever:~
$ mysql -uroot data
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.23 Homebrew

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

root@localhost [data]>|
```

### PS1 personnalisé via le fichier de configuration MySQL

Dans `mysqld.cnf` ou équivalent:

```
[mysql]
prompt = '\u@\h [\d]> '
```

Cela produit un effet similaire, sans avoir à traiter avec `.bashrc`.

Lire Personnaliser PS1 en ligne: <https://riptutorial.com/fr/mysql/topic/5795/personnaliser-ps1>

---

# Chapitre 49: Pointes

## Exemples

### Utilisation de backticks

Il existe de nombreux exemples où sont utilisés dans les apostrophes inverses d' une requête mais pour beaucoup il est encore difficile de savoir quand ou où utiliser des accents graves `` ` ` .

Les backticks sont principalement utilisés pour empêcher une erreur appelée " *mot réservé MySQL* ". Lorsque vous créez une table dans PHPmyAdmin, vous êtes parfois confronté à un avertissement ou à une alerte indiquant que vous utilisez un " *mot réservé MySQL* ".

Par exemple, lorsque vous créez une table avec une colonne nommée " `group` ", vous obtenez un avertissement. En effet, vous pouvez effectuer la requête suivante:

```
SELECT student_name, AVG(test_score) FROM student GROUP BY group
```

Pour vous assurer de ne pas avoir d'erreur dans votre requête, vous devez utiliser des backticks afin que votre requête devienne:

```
SELECT student_name, AVG(test_score) FROM student GROUP BY `group`
```

### Table

---

Non seulement les noms de colonnes peuvent être entourés de backticks, mais aussi de noms de tables. Par exemple, lorsque vous devez `JOIN` plusieurs tables.

```
SELECT `users`.`username`, `groups`.`group` FROM `users`
```

### Plus facile à lire

---

Comme vous pouvez le voir, l'utilisation de backticks autour des noms de tables et de colonnes facilite également la lecture de la requête.

Par exemple, lorsque vous écrivez des requêtes toutes en minuscule:

```
select student_name, AVG(test_score) from student group by group
select `student_name`, AVG(`test_score`) from `student` group by `group`
```

Veillez consulter la page du manuel MySQL intitulée [Mots-clés et mots réservés](#) . Ceux avec un (R) sont des mots réservés. Les autres ne sont que des mots-clés. Les réservées exigent une attention particulière.

Lire Pointes en ligne: <https://riptutorial.com/fr/mysql/topic/5208/pointes>

# Chapitre 50: PREPARE Déclarations

## Syntaxe

- PREPARE stmt\_name FROM preparable\_stmt
- EXECUTE nom\_stmt [USING @var\_name [, @var\_name] ...]
- {DEALLOCATE | DROP} PREPARE stmt\_name

## Exemples

### PREPARE, EXECUTE et DEALLOCATE PREPARE Déclarations

**PREPARE** prépare une déclaration pour exécution

**EXECUTE** exécute une déclaration préparée

**DEALLOCATE PREPARE** publie une déclaration préparée

```
SET @s = 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';
PREPARE stmt2 FROM @s;
SET @a = 6;
SET @b = 8;
EXECUTE stmt2 USING @a, @b;
```

Résultat:

```
+-----+
| hypotenuse |
+-----+
|          10 |
+-----+
```

Finalement,

```
DEALLOCATE PREPARE stmt2;
```

Remarques:

- Vous devez utiliser @variables, pas les variables DECLARED pour FROM @s
- Une utilisation principale de Prepare, etc. consiste à "construire" une requête pour les situations où la liaison ne fonctionnera pas, comme l'insertion du nom de la table.

## Construire et exécuter

(Ceci est une demande pour un bon exemple qui montre comment *construire* un `SELECT` utilisant `CONCAT`, puis préparer + exécuter. S'il vous plaît insister sur l'utilisation de @variables contre les variables DECLARED - cela fait une grande différence, et c'est quelque chose que les novices (



inclure moi-même) trébucher.)

## Alter table avec add column

```
SET v_column_definition := CONCAT(  
    v_column_name  
    , ' ', v_column_type  
    , ' ', v_column_options  
);  
  
SET @stmt := CONCAT('ALTER TABLE ADD COLUMN ', v_column_definition);  
  
PREPARE stmt FROM @stmt;  
EXECUTE stmt;  
DEALLOCATE PREPARE stmt;
```

Lire **PREPARE Déclarations en ligne**: <https://riptutorial.com/fr/mysql/topic/2603/prepare-declarations>

# Chapitre 51: Recherche en texte intégral

## Introduction

MySQL offre la recherche FULLTEXT. Il recherche les tables avec des colonnes contenant du texte pour les meilleures correspondances pour les mots et les phrases.

## Remarques

FULLTEXT recherche FULLTEXT fonctionne étrangement sur les tables contenant un petit nombre de lignes. Ainsi, lorsque vous en faites l'essai, il peut être utile de vous procurer une table de taille moyenne en ligne. Voici un [tableau des éléments du livre](#), avec des titres et des auteurs. Vous pouvez le télécharger, le décompresser et le charger dans MySQL.

FULLTEXT recherche FULLTEXT est destinée à être utilisée avec une assistance humaine. Il est conçu pour produire plus de correspondances qu'une opération de filtrage WHERE column LIKE 'text%' .

FULLTEXT recherche FULLTEXT est disponible pour les tables MyISAM . Il est également disponible pour les tables InnoDB dans MySQL version 5.6.4 ou ultérieure.

## Exemples

### Recherche FULLTEXT simple

```
SET @searchTerm= 'Database Programming';
SELECT MATCH (Title) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) Score,
       ISBN, Author, Title
FROM book
WHERE MATCH (Title) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE)
ORDER BY MATCH (Title) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) DESC;
```

Étant donné une table nommée `book` avec des colonnes nommées `ISBN`, «Titre» et «Auteur», cela permet de trouver des livres correspondant aux termes 'Database Programming' . Il montre les meilleurs matchs en premier.

Pour que cela fonctionne, un index de texte intégral sur la colonne `Title` doit être disponible:

```
ALTER TABLE book ADD FULLTEXT INDEX Fulltext_title_index (Title);
```

### Recherche BOOLEAN simple

```
SET @searchTerm= 'Database Programming -Java';
SELECT MATCH (Title) AGAINST (@searchTerm IN BOOLEAN MODE) Score,
       ISBN, Author, Title
FROM book
WHERE MATCH (Title) AGAINST (@searchTerm IN BOOLEAN MODE)
```

```
ORDER BY MATCH (Title) AGAINST (@searchTerm IN BOOLEAN MODE) DESC;
```

Étant donné un tableau nommé `book` avec des colonnes nommées `ISBN`, `Title` et `Author`, cette recherche recherche les livres contenant les mots 'Database' et 'Programming' dans le titre, mais pas le mot 'Java'.

Pour que cela fonctionne, un index de texte intégral sur la colonne Titre doit être disponible:

```
ALTER TABLE book ADD FULLTEXT INDEX Fulltext_title_index (Title);
```

## Recherche FULLTEXT sur plusieurs colonnes

```
SET @searchTerm= 'Date Database Programming';
SELECT MATCH (Title, Author) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) Score,
       ISBN, Author, Title
FROM book
WHERE MATCH (Title, Author) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE)
ORDER BY MATCH (Title, Author) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) DESC;
```

Étant donné un tableau nommé `livre` avec des colonnes nommées `ISBN`, `Title` et `Author`, les livres correspondant aux termes "Programmation de la base de données de dates" sont trouvés. Il montre les meilleurs matchs en premier. Les meilleurs résultats incluent des livres écrits par le Prof. CJ Date.

(Mais l'un des meilleurs résultats est également *le Guide de datation du Docteur Date: Comment passer du premier rendez-vous au parfait compagnon*. Cela montre une limitation de la recherche FULLTEXT: il ne prétend pas comprendre des choses telles que des parties du discours ou la signification des mots indexés.)

Pour que cela fonctionne, un index de texte intégral sur les colonnes Title et Author doit être disponible:

```
ALTER TABLE book ADD FULLTEXT INDEX Fulltext_title_author_index (Title, Author);
```

Lire Recherche en texte intégral en ligne: <https://riptutorial.com/fr/mysql/topic/8759/recherche-en-texte-integral>

---

# Chapitre 52: Récupérer du mot de passe root perdu

## Exemples

Définir le mot de passe root, activer l'utilisateur root pour l'accès socket et http

Résout le problème de: accès refusé pour l'utilisateur root en utilisant le mot de passe YES Arrête mySQL:

```
sudo systemctl stop mysql
```

Redémarrez mySQL, en ignorant les tables de droits:

```
sudo mysqld_safe --skip-grant-tables
```

S'identifier:

```
mysql -u root
```

Dans le shell SQL, regardez si les utilisateurs existent:

```
select User, password,plugin FROM mysql.user ;
```

Mettre à jour les utilisateurs (plugin null permet à tous les plugins):

```
update mysql.user set password=PASSWORD('mypassword'), plugin = NULL WHERE User = 'root';  
exit;
```

Dans le shell Unix, arrêtez mySQL sans les tables de droits, puis redémarrez avec les tables de droits:

```
sudo service mysql stop  
sudo service mysql start
```

Lire Récupérer du mot de passe root perdu en ligne:

<https://riptutorial.com/fr/mysql/topic/9973/recuperer-du-mot-de-passe-root-perdu>

---

# Chapitre 53: Récupérer et réinitialiser le mot de passe root par défaut pour MySQL 5.7+

## Introduction

Après MySQL 5.7, lorsque nous installons MySQL, nous n'avons parfois pas besoin de créer un compte root ou de donner un mot de passe root. Par défaut, lorsque nous démarrons le serveur, le mot de passe par défaut est stocké dans le fichier `mysqld.log`. Nous devons nous connecter au système en utilisant ce mot de passe et nous devons le changer.

## Remarques

La récupération et la réinitialisation du mot de passe root par défaut à l'aide de cette méthode est applicable uniquement à MySQL 5.7+

## Exemples

### Que se passe-t-il lorsque le démarrage initial du serveur

Étant donné que le répertoire de données du serveur est vide:

- Le serveur est initialisé.
- Le certificat SSL et les fichiers de clés sont générés dans le répertoire de données.
- Le plugin `validate_password` est installé et activé.
- Le compte superutilisateur 'root' @ 'localhost' est créé. Le mot de passe du superutilisateur est défini et stocké dans le fichier journal des erreurs.

### Comment changer le mot de passe root en utilisant le mot de passe par défaut

Pour révéler le mot de passe "root" par défaut:

```
shell> sudo grep 'temporary password' /var/log/mysqld.log
```

Modifiez le mot de passe root dès que possible en vous connectant avec le mot de passe temporaire généré et définissez un mot de passe personnalisé pour le compte superutilisateur:

```
shell> mysql -uroot -p
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass5!';
```

**Remarque:** le plug-in `validate_password` de MySQL est installé par défaut. Cela nécessitera que les mots de passe contiennent au moins une lettre majuscule, une lettre minuscule, un chiffre et un caractère spécial, et que la longueur totale du mot de passe soit d'au moins 8 caractères.

## réinitialiser le mot de passe root lorsque "/ var / run / mysqld" pour le fichier de socket UNIX n'existe pas "

si j'oublie le mot de passe, j'aurai une erreur.

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

Entrer le mot de passe:

ERROR 1045 (28000): Accès refusé pour l'utilisateur 'root' @ 'localhost' (à l'aide du mot de passe: YES)

J'ai essayé de résoudre le problème en connaissant d'abord le statut:

```
$ systemctl status mysql.service
```

mysql.service - Serveur de communauté MySQL chargé: chargé  
(/lib/systemd/system/mysql.service; activé; paramètre prédéfini du fournisseur: en Actif: actif (en cours d'exécution) depuis le jeu. 2017-06-08 14:31:33 IST; il y a 38s

Ensuite, j'ai utilisé le code `mysqld_safe --skip-grant-tables &` mais j'ai eu l'erreur:

Le répertoire `mysqld_safe / var / run / mysqld` pour le fichier de socket UNIX n'existe pas.

```
$ systemctl stop mysql.service  
$ ps -eaf|grep mysql  
$ mysqld_safe --skip-grant-tables &
```

J'ai résolu:

```
$ mkdir -p /var/run/mysqld  
$ chown mysql:mysql /var/run/mysqld
```

Maintenant, j'utilise le même code `mysqld_safe --skip-grant-tables &` et

`mysqld_safe` Démarrage du démon `mysqld` avec les bases de données de `/ var / lib / mysql`

Si j'utilise `$ mysql -u root` j'aurai:

Version du serveur: 5.7.18-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle et / ou ses filiales. Tous les droits sont réservés.

Oracle est une marque déposée d'Oracle Corporation et / ou de ses filiales. Les autres noms peuvent être des marques déposées de leurs propriétaires respectifs.

Tapez 'help;' ou '\ h' pour de l'aide. Tapez '\ c' pour effacer la déclaration en cours.

mysql>

Il est maintenant temps de changer le mot de passe:

```
mysql> use mysql
mysql> describe user;
```

Lire les informations de la table pour compléter les noms de table et de colonne Vous pouvez désactiver cette fonction pour obtenir un démarrage plus rapide avec -A

Base de données modifiée

```
mysql> FLUSH PRIVILEGES;
mysql> SET PASSWORD FOR root@'localhost' = PASSWORD('newpwd');
```

ou Si vous avez un compte root mysql qui peut se connecter de partout, vous devriez aussi faire:

```
UPDATE mysql.user SET Password=PASSWORD('newpwd') WHERE User='root';
```

Méthode alternative:

```
USE mysql
UPDATE user SET Password = PASSWORD('newpwd')
WHERE Host = 'localhost' AND User = 'root';
```

Et si vous avez un compte root qui peut accéder de partout:

```
USE mysql
UPDATE user SET Password = PASSWORD('newpwd')
WHERE Host = '%' AND User = 'root';`enter code here
```

maintenant besoin de `quit` mysql et stop / start

```
FLUSH PRIVILEGES;
sudo /etc/init.d/mysql stop
sudo /etc/init.d/mysql start
```

maintenant à nouveau ``mysql -u root -p` et utiliser le nouveau mot de passe pour obtenir

mysql>

Lire [Récupérer et réinitialiser le mot de passe root par défaut pour MySQL 5.7+ en ligne: https://riptutorial.com/fr/mysql/topic/9563/recuperer-et-reinitialiser-le-mot-de-passe-root-par-defaut-pour-mysql-5-7plus](https://riptutorial.com/fr/mysql/topic/9563/recuperer-et-reinitialiser-le-mot-de-passe-root-par-defaut-pour-mysql-5-7plus)

---

# Chapitre 54: Réplication

## Remarques

La réplication est utilisée pour copier des données [de sauvegarde] depuis un serveur de base de données MySQL vers un ou plusieurs serveurs de base de données MySQL.

**Master** - Le serveur de base de données MySQL, qui sert les données à copier

**Slave** - Le serveur de base de données MySQL copie les données servies par Master

Avec MySQL, la réplication est asynchrone par défaut. Cela signifie que les esclaves n'ont pas besoin d'être connectés en permanence pour recevoir les mises à jour du maître. Par exemple, si votre esclave est éteint ou n'est pas connecté au maître et que vous changez d'esclave ou que vous vous connectez ultérieurement au maître, il se synchronisera automatiquement avec le maître.

Selon la configuration, vous pouvez répliquer toutes les bases de données, les bases de données sélectionnées ou même les tables sélectionnées dans une base de données.

## Formats de réplication

Il existe deux types principaux de formats de réplication

*Statement Based Replication (SBR)* - réplique des instructions SQL entières. Dans ce cas, le maître écrit des instructions SQL dans le journal binaire. La réplication du maître sur l'esclave fonctionne en exécutant les instructions SQL sur l'esclave.

*Row Based Replication (RBR)* - réplique uniquement les lignes modifiées. Dans ce cas, le maître écrit des événements dans le journal binaire pour indiquer comment les lignes de la table sont modifiées. La réplication du maître sur l'esclave fonctionne en copiant les événements représentant les modifications apportées aux lignes de la table à l'esclave.

Vous pouvez également utiliser une troisième variété, *MBR (Mixed Based Replication)*. Dans ce cas, la journalisation basée sur l'instruction et sur la ligne est utilisée. Le journal sera créé en fonction de ce qui est le plus approprié pour le changement.

Le format basé sur les instructions était le format par défaut dans les versions de MySQL antérieures à 5.7.7. En MySQL 5.7.7 et versions ultérieures, le format basé sur les lignes est le format par défaut.

## Exemples

### Maître - Configuration de la réplication esclave

Considérons 2 serveurs MySQL pour la configuration de la réplication, l'un étant un maître et l'autre un esclave.



Nous allons configurer le maître pour qu'il conserve un journal de chaque action effectuée sur celui-ci. Nous allons configurer le serveur esclave qu'il devrait regarder le journal sur le maître et chaque fois que des changements se produisent dans le journal sur le maître, il devrait faire la même chose.

## **Configuration principale**

Tout d'abord, nous devons créer un utilisateur sur le maître. Cet utilisateur va être utilisé par Slave pour créer une connexion avec le maître.

```
CREATE USER 'user_name'@'%' IDENTIFIED BY 'user_password';
GRANT REPLICATION SLAVE ON *.* TO 'user_name'@'%';
FLUSH PRIVILEGES;
```

Modifiez `user_name` et `user_password` fonction de votre nom d'utilisateur et de votre mot de passe.

Maintenant, le `my.inf` (`my.cnf` sous Linux) doit être modifié. Incluez les lignes suivantes dans la section `[mysqld]`.

```
server-id = 1
log-bin = mysql-bin.log
binlog-do-db = your_database
```

La première ligne est utilisée pour attribuer un identifiant à ce serveur MySQL.

La deuxième ligne indique à MySQL de commencer à écrire un journal dans le fichier journal spécifié. Sous Linux, cela peut être configuré comme `log-bin = /home/mysql/logs/mysql-bin.log`. Si vous commencez la réplication sur un serveur MySQL dans lequel la réplication a déjà été utilisée, assurez-vous que ce répertoire est vide de tous les journaux de réplication.

La troisième ligne est utilisée pour configurer la base de données pour laquelle nous allons écrire le journal. Vous devez remplacer `your_database` par le nom de votre base de données.

Assurez-vous que `skip-networking` n'a pas été activé et redémarrez le serveur MySQL (Master)

## **Configuration d'esclave**

Le `my.inf` fichier `my.inf` doit également être édité dans Slave. Incluez les lignes suivantes dans la section `[mysqld]`.

```
server-id = 2
master-host = master_ip_address
master-connect-retry = 60

master-user = user_name
master-password = user_password
replicate-do-db = your_database

relay-log = slave-relay.log
relay-log-index = slave-relay-log.index
```

La première ligne est utilisée pour attribuer un identifiant à ce serveur MySQL. Cet identifiant doit être unique.

La deuxième ligne est l'adresse IP du serveur maître. Changez cela en fonction de l'IP de votre système maître

La troisième ligne est utilisée pour définir une limite de relance en secondes.

Les deux lignes suivantes indiquent le nom d'utilisateur et le mot de passe de l'esclave, en utilisant le lien avec le maître.

La ligne suivante définit la base de données à répliquer.

Les deux dernières lignes utilisées pour attribuer des noms de fichier `relay-log` et `relay-log-index`.

Assurez-vous que le `skip-networking` n'a pas été activé et redémarrez le serveur MySQL (esclave)

### ***Copier des données sur un esclave***

Si des données sont constamment ajoutées au maître, nous devons empêcher tout accès à la base de données sur le maître afin que rien ne puisse être ajouté. Cela peut être réalisé en exécutant l'instruction suivante dans Master.

```
FLUSH TABLES WITH READ LOCK;
```

Si aucune donnée n'est ajoutée au serveur, vous pouvez ignorer l'étape ci-dessus.

Nous allons prendre la sauvegarde des données du maître en utilisant `mysqldump`

```
mysqldump your_database -u root -p > D://Backup/backup.sql;
```

Modifiez `your_database` répertoire de base de `your_database` et de sauvegarde en fonction de votre configuration. Vous aurez maintenant un fichier appelé `backup.sql` à l'emplacement indiqué.

Si votre base de données n'existe pas dans votre esclave, créez-la en exécutant ce qui suit

```
CREATE DATABASE `your_database`;
```

Maintenant, nous devons importer la sauvegarde dans le serveur MySQL Slave.

```
mysql -u root -p your_database <D://Backup/backup.sql  
--->Change `your_database` and backup directory according to your setup
```

### ***Démarrer la réplication***

Pour démarrer la réplication, il est nécessaire de trouver le nom du fichier journal et la position du journal dans le maître. Donc, lancez ce qui suit dans Master

```
SHOW MASTER STATUS;
```

Cela vous donnera une sortie comme ci-dessous

```
+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB   | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-bin.000001 | 130      | your_database  |                    |
+-----+-----+-----+-----+
```

Ensuite, exécutez ce qui suit dans Slave

```
SLAVE STOP;
CHANGE MASTER TO MASTER_HOST='master_ip_address', MASTER_USER='user_name',
    MASTER_PASSWORD='user_password', MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=130;
SLAVE START;
```

D'abord, nous arrêtons l'esclave. Ensuite, nous lui disons exactement où regarder dans le fichier journal maître. Pour le nom `MASTER_LOG_FILE` et `MASTER_LOG_POS`, utilisez les valeurs obtenues en exécutant la commande `SHOW MASTER STATUS` sur le maître.

Vous devez modifier l'adresse IP du maître dans `MASTER_HOST` et modifier l'utilisateur et le mot de passe en conséquence.

L'esclave va maintenant attendre. L'état de l'esclave peut être consulté en exécutant ce qui suit

```
SHOW SLAVE STATUS;
```

Si vous avez précédemment exécuté `FLUSH TABLES WITH READ LOCK` dans Master, libérez les tables de lock by run de la manière suivante:

```
UNLOCK TABLES;
```

Maintenant, le maître garde un journal pour chaque action effectuée et le serveur esclave examine le journal du maître. Chaque fois que des changements se produisent dans le journal sur le maître, Slave réplique cela.

## Erreurs de réplication

Chaque fois qu'il y a une erreur lors de l'exécution d'une requête sur l'esclave, MySQL arrête automatiquement la réplication pour identifier le problème et le corriger. Cela est principalement dû au fait qu'un événement a provoqué une duplication de la clé ou qu'une ligne n'a pas été trouvée et qu'elle ne peut être ni mise à jour ni supprimée. Vous pouvez ignorer ces erreurs, même si cela n'est pas recommandé

Pour ignorer une seule requête qui suspend l'esclave, utilisez la syntaxe suivante

```
SET GLOBAL sql_slave_skip_counter = N;
```

Cette instruction ignore les N prochains événements du maître. Cette instruction est valide uniquement lorsque les threads esclaves ne sont pas en cours d'exécution. Sinon, il génère une

erreur.

```
STOP SLAVE;  
SET GLOBAL sql_slave_skip_counter=1;  
START SLAVE;
```

Dans certains cas, cela va bien. Mais si l'instruction fait partie d'une transaction à plusieurs instructions, elle devient plus complexe, car ignorer l'instruction générant l'erreur provoquera l'ignition de la transaction entière.

Si vous voulez ignorer plus de requêtes produisant le même code d'erreur et si vous êtes sûr que sauter ces erreurs ne rendra pas votre esclave incohérent et que vous voulez tout ignorer, vous ajouterez une ligne pour ignorer ce code d'erreur dans `my.cnf`.

Par exemple, vous pouvez ignorer toutes les erreurs en double que vous pourriez avoir

```
1062 | Error 'Duplicate entry 'xyz' for key 1' on query
```

Ensuite, ajoutez ce qui suit à votre `my.cnf`

```
slave-skip-errors = 1062
```

Vous pouvez également ignorer d'autres types d'erreurs ou tous les codes d'erreur, mais veillez à ne pas ignorer ces erreurs si vous ignorez ces erreurs. Voici la syntaxe et les exemples

```
slave-skip-errors=[err_code1,err_code2,...|all]  
  
slave-skip-errors=1062,1053  
slave-skip-errors=all  
slave-skip-errors=ddl_exist_errors
```

Lire Réplication en ligne: <https://riptutorial.com/fr/mysql/topic/7218/replication>

# Chapitre 55: Requêtes pivot

## Remarques

La création de requête de pivot dans MySQL repose sur la fonction `GROUP_CONCAT()`. Si le résultat de l'expression qui crée les colonnes de la requête pivot doit être élevé, la valeur de la variable `group_concat_max_len` doit être augmentée:

```
set session group_concat_max_len = 1024 * 1024; -- This should be enough for most cases
```

## Exemples

### Création d'une requête pivot

MySQL ne fournit pas de moyen intégré pour créer des requêtes pivot. Cependant, ceux-ci peuvent être créés à l'aide d'instructions préparées.

Supposons la table `tbl_values` :

| Id | prénom | Groupe | Valeur |
|----|--------|--------|--------|
| 1  | Pete   | UNE    | dix    |
| 2  | Pete   | B      | 20     |
| 3  | John   | UNE    | dix    |

Request: crée une requête qui affiche la somme de `Value` pour chaque `Name` ; le `Group` doit être un en-tête de colonne et le `Name` doit être l'en-tête de la ligne.

```
-- 1. Create an expression that builds the columns
set @sql = (
  select group_concat(distinct
    concat(
      "sum(case when `Group`='", Group, "' then `Value` end) as `", `Group`, "`"
    )
  )
  from tbl_values
);

-- 2. Complete the SQL instruction
set @sql = concat("select Name, ", @sql, " from tbl_values group by `Name`");

-- 3. Create a prepared statement
prepare stmt from @sql;

-- 4. Execute the prepared statement
execute stmt;
```

Résultat:

| prénom | UNE | B   |
|--------|-----|-----|
| John   | dix | NUL |
| Pete   | dix | 20  |

**Important:** libérez l'instruction préparée une fois qu'elle n'est plus nécessaire:

```
deallocate prepare stmt;
```

[Exemple sur SQL Fiddle](#)

Lire [Requêtes pivot en ligne](https://riptutorial.com/fr/mysql/topic/3074/requetes-pivot): <https://riptutorial.com/fr/mysql/topic/3074/requetes-pivot>

# Chapitre 56: Routines stockées (procédures et fonctions)

## Paramètres

| Paramètre | Détails  |
|-----------|--|
| RÉSULTATS | Spécifie le type de données pouvant être renvoyé par une fonction.   |
| REVENIR   | La variable ou la valeur réelle qui suit la syntaxe <code>RETURN</code> à la valeur renvoyée à partir de laquelle la fonction a été appelée. |

## Remarques

Une routine stockée est une procédure ou une fonction.

Une procédure est appelée à l'aide d'une instruction `CALL` et ne peut que renvoyer des valeurs à l'aide de variables de sortie.

Une fonction peut être appelée depuis une instruction comme toute autre fonction et peut renvoyer une valeur scalaire.

## Exemples

### Créer une fonction

La fonction exemple (triviale) suivante renvoie simplement la valeur `INT` constante `12` .

```
DELIMITER ||
CREATE FUNCTION fonctionname ()
RETURNS INT
BEGIN
    RETURN 12;
END;
||
DELIMITER ;
```

La première ligne définit ce que le caractère de délimiteur ( `DELIMITER ||` ) doit être changé, il faut le définir avant de créer une fonction, sinon si elle est laissée à sa valeur par défaut ; puis le premier ; ce qui se trouve dans le corps de la fonction sera considéré comme la fin de l'instruction `CREATE` , ce qui n'est généralement pas ce que l'on souhaite.

Après l'exécution de `CREATE FUNCTION` , vous devez redéfinir le délimiteur sur sa valeur par défaut ; comme on le voit après le code de fonction dans l'exemple ci-dessus ( `DELIMITER ;` ).

Exécution cette fonction est la suivante:

```
SELECT fonctionname();
+-----+
| fonctionname() |
+-----+
|           12 |
+-----+
```

Un exemple légèrement plus complexe (mais néanmoins trivial) prend un paramètre et lui ajoute une constante:

```
DELIMITER $$
CREATE FUNCTION add_2 ( my_arg INT )
  RETURNS INT
BEGIN
  RETURN (my_arg + 2);
END;
$$
DELIMITER ;

SELECT add_2(12);
+-----+
| add_2(12) |
+-----+
|           14 |
+-----+
```

Notez l'utilisation d'un argument différent pour la directive `DELIMITER`. Vous pouvez réellement utiliser n'importe quelle séquence de caractères n'apparaissant pas dans le corps de l'instruction `CREATE`, mais la pratique habituelle consiste à utiliser un caractère non alphanumérique doublé tel que `\\`, `||` ou `$$`.

Il est recommandé de toujours modifier le paramètre avant et après une fonction, une procédure ou une création ou une mise à jour de déclencheur car certaines interfaces graphiques ne nécessitent pas de modification du délimiteur alors que l'exécution de requêtes via la ligne de commande nécessite toujours la définition du délimiteur.

## Créer une procédure avec une préparation construite

```
DROP PROCEDURE if exists displayNext100WithName;
DELIMITER $$
CREATE PROCEDURE displayNext100WithName
(
  nStart int,
  tblName varchar(100)
)
BEGIN
  DECLARE thesql varchar(500); -- holds the constructed sql string to execute

  -- expands the sizing of the output buffer to accomodate the output (Max value is at least 4GB)
  SET session group_concat_max_len = 4096; -- prevents group_concat from barfing with error 1160 or whatever it is
```



```

SET @thesql=CONCAT("select group_concat(qid order by qid SEPARATOR '%3B') as nums ","from
(
  select qid from ");
SET @thesql=CONCAT(@thesql,tblName," where qid>? order by qid limit 100 )xDerived");
PREPARE stmt1 FROM @thesql; -- create a statement object from the construct sql string to
execute
SET @p1 = nStart; -- transfers parameter passed into a User Variable compatible with the
below EXECUTE
EXECUTE stmt1 USING @p1;

DEALLOCATE PREPARE stmt1; -- deallocate the statement object when finished
END$$
DELIMITER ;

```

La création de la procédure stockée montre l'encapsulation avec un DELIMITER nécessaire dans de nombreux outils clients.

Exemple d'appel:

```
call displayNext100WithName(1,"questions_mysql");
```

Exemple de sortie avec le séparateur %3B (point-virgule):

| nums  |
|---|
| 607264%;3B20173649%;3B30532900%;3B32030116%;3B32145357%;3B32166934%;3B32298065%;3B32793619%;3B333210... |

## Procédure stockée avec les paramètres IN, OUT, INOUT

```

DELIMITER $$

DROP PROCEDURE IF EXISTS sp_nested_loop$$
CREATE PROCEDURE sp_nested_loop(IN i INT, IN j INT, OUT x INT, OUT y INT, INOUT z INT)
BEGIN
  DECLARE a INTEGER DEFAULT 0;
  DECLARE b INTEGER DEFAULT 0;
  DECLARE c INTEGER DEFAULT 0;
  WHILE a < i DO
    WHILE b < j DO
      SET c = c + 1;
      SET b = b + 1;
    END WHILE;
    SET a = a + 1;
    SET b = 0;
  END WHILE;
  SET x = a, y = c;
  SET z = x + y + z;
END $$
DELIMITER ;

```

Invoke ( **CALL** ) la procédure stockée:

```

SET @z = 30;
call sp_nested_loop(10, 20, @x, @y, @z);
SELECT @x, @y, @z;

```

Résultat:

```
+-----+-----+-----+
|  @x  |  @y  |  @z  |
+-----+-----+-----+
|  10  |  200 |  240 |
+-----+-----+-----+
```

Un paramètre `IN` transmet une valeur à une procédure. La procédure peut modifier la valeur, mais la modification n'est pas visible pour l'appelant au retour de la procédure.

Un paramètre `OUT` renvoie une valeur de la procédure à l'appelant. Sa valeur initiale est `NULL` dans la procédure et sa valeur est visible pour l'appelant lors du retour de la procédure.

Un paramètre `INOUT` est initialisé par l'appelant, peut être modifié par la procédure et toute modification apportée par la procédure est visible par l'appelant au retour de la procédure.

Réf: <http://dev.mysql.com/doc/refman/5.7/fr/create-procedure.html>

## Des curseurs

Les curseurs vous permettent d'itérer les résultats de la requête une par une. `DECLARE` commande `DECLARE` est utilisée pour initialiser le curseur et l'associer à une requête SQL spécifique:

```
DECLARE student CURSOR FOR SELECT name FROM student;
```

Disons que nous vendons des produits de certains types. Nous voulons compter le nombre de produits de chaque type.

Nos données:

```
CREATE TABLE product
(
  id    INT(10) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  type  VARCHAR(50)      NOT NULL,
  name  VARCHAR(255)     NOT NULL
);
CREATE TABLE product_type
(
  name VARCHAR(50) NOT NULL PRIMARY KEY
);
CREATE TABLE product_type_count
(
  type  VARCHAR(50)      NOT NULL PRIMARY KEY,
  count INT(10) UNSIGNED NOT NULL DEFAULT 0
);
INSERT INTO product_type (name) VALUES
('dress'),
('food');
INSERT INTO product (type, name) VALUES
('dress', 'T-shirt'),
('dress', 'Trousers'),
('food', 'Apple'),
```

```
('food', 'Tomatoes'),
('food', 'Meat');
```

Nous pouvons atteindre l'objectif en utilisant la procédure stockée à l'aide du curseur:

```
DELIMITER //
DROP PROCEDURE IF EXISTS product_count;
CREATE PROCEDURE product_count()
BEGIN
    DECLARE p_type VARCHAR(255);
    DECLARE p_count INT(10) UNSIGNED;
    DECLARE done INT DEFAULT 0;
    DECLARE product CURSOR FOR
        SELECT
            type,
            COUNT(*)
        FROM product
        GROUP BY type;
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;

    TRUNCATE product_type;

    OPEN product;

    REPEAT
        FETCH product
        INTO p_type, p_count;
        IF NOT done
        THEN
            INSERT INTO product_type_count
            SET
                type = p_type,
                count = p_count;
        END IF;
    UNTIL done
    END REPEAT;

    CLOSE product;
END //
DELIMITER ;
```

Quand vous pouvez appeler la procédure avec:

```
CALL product_count();
```

Le résultat serait dans la table `product_type_count` :

| type  | count |
|-------|-------|
| dress | 2     |
| food  | 3     |

Bien que ce soit un bon exemple de `CURSOR` , notez comment le corps entier de la procédure peut être remplacé par un simple

```
INSERT INTO product_type_count
```

```
(type, count)
SELECT type, COUNT(*)
FROM product
GROUP BY type;
```

Cela va courir beaucoup plus vite.

## Plusieurs ResultSets

Contrairement à une `SELECT`, une `Stored Procedure` renvoie plusieurs jeux de résultats. Le code différent requis pour être utilisé pour rassembler les résultats d'un `CALL` en Perl, PHP, etc.

(Besoin d'un code spécifique ici ou ailleurs!)

## Créer une fonction

```
DELIMITER $$
CREATE
  DEFINER=`db_username`@`hostname_or_IP`
  FUNCTION `function_name` (optional_param data_type(length_if_applicable))
  RETURNS data_type
BEGIN
  /*
  SQL Statements goes here
  */
END$$
DELIMITER ;
```

Le type de données `RETURNS` est tout type de données MySQL.

[Lire Routines stockées \(procédures et fonctions\) en ligne:](#)

<https://riptutorial.com/fr/mysql/topic/1351/routines-stockees--procedures-et-fonctions->

# Chapitre 57: Sauvegarde avec mysqldump

## Syntaxe

- `mysqldump -u [nom d'utilisateur] -p [mot de passe] [autres options] nom_base> dumpFileName.sql ///` Pour sauvegarder une base de données unique
- `mysqldump -u [nom d'utilisateur] -p [mot de passe] [autres options] nom_base [nom_de_table1 nom_de_table2 nom_de_table2 ...]> dumpFileName.sql ///` Pour sauvegarder une ou plusieurs tables
- `mysqldump -u [nom d'utilisateur] -p [mot de passe] [autres options] --databases nom_base1 nom_base2 nom_base3 ...> nom_du_fichier.sql ///` Pour sauvegarder une ou plusieurs bases de données complètes
- `mysqldump -u [nom d'utilisateur] -p [mot de passe] [autres options] --all-databases> dumpFileName.sql ///` Pour sauvegarder tout le serveur MySQL

## Paramètres

| Option                  | Effet   |
|-------------------------|---|
| -                       | # Options de connexion au serveur   |
| -h ( --host )           | Hôte (adresse IP ou nom d'hôte) auquel se connecter. La valeur par défaut est localhost ( 127.0.0.1 ) Exemple: -h localhost   |
| -u ( --user )           | Utilisateur MySQL   |
| -p ( --password )       | Mot de passe MySQL. <b>Important</b> : Lorsque vous utilisez -p , il ne doit pas y avoir d'espace entre l'option et le mot de passe. Exemple: -pMyPassword  |
| -                       | # Options de vidage   |
| --add-drop-database     | Ajoutez une instruction <code>DROP DATABASE</code> avant chaque instruction <code>CREATE DATABASE</code> . Utile si vous souhaitez remplacer des bases de données sur le serveur.   |
| --add-drop-table        | Ajoutez une instruction <code>DROP TABLE</code> avant chaque instruction <code>CREATE TABLE</code> . Utile si vous souhaitez remplacer des tables sur le serveur.   |
| --no-create-db          | Supprimez les instructions <code>CREATE DATABASE</code> dans le vidage. Ceci est utile lorsque vous êtes sûr que la ou les bases de données que vous mettez en place existent déjà sur le serveur où vous allez charger le fichier de vidage.                           |
| -t ( --no-create-info ) | Supprimez toutes les instructions <code>CREATE TABLE</code> dans le vidage. Ceci est utile lorsque vous souhaitez vider uniquement les données des tables et utiliser le fichier de vidage pour remplir des tables identiques dans une autre base de données / serveur. |

| Option                             | Effet   |
|------------------------------------|---|
| <code>-d ( --no-data )</code>      | Ne pas écrire d'informations de table. Cela ne videra que les instructions <code>CREATE TABLE</code> . Utile pour créer des bases de données "template"   |
| <code>-R ( --routines )</code>     | Inclure les procédures / fonctions stockées dans le vidage.   |
| <code>-K ( --disable-keys )</code> | Désactivez les clés pour chaque table avant d'insérer les données et activez les clés une fois les données insérées. Cela accélère les insertions uniquement dans les tables MyISAM avec des index non uniques. |

## Remarques

La sortie d'une opération `mysqldump` est un fichier légèrement commenté contenant des instructions SQL séquentielles compatibles avec la version des utilitaires MySQL utilisés pour le générer (avec une attention particulière portée sur la compatibilité avec les versions précédentes, mais aucune garantie pour les versions futures). Ainsi, la restauration d'une base de données `mysqldump ed` comprend l'exécution de ces instructions. Généralement, ce fichier

- `DROP` s la première table ou vue spécifiée
- `CREATE` s cette table ou vue
- Pour les tables vidées avec des données (c'est-à-dire sans l'option `--no-data` )
  - `LOCK` s la table
  - `INSERT` s toutes les lignes de la table d'origine dans une seule déclaration
- `UNLOCK TABLES`
- Répète ce qui précède pour toutes les autres tables et vues
- `DROP` s la première routine incluse
- `CREATE` cette routine
- Répète la même chose pour toutes les autres routines

La présence de `DROP` avant `CREATE` pour chaque table signifie que si le schéma est présent, qu'il soit vide ou non, l'utilisation d'un fichier `mysqldump` pour sa restauration remplira ou écrasera les données.

## Exemples

### Création d'une sauvegarde d'une base de données ou d'une table

Créez un instantané de toute une base de données:

```
mysqldump [options] db_name > filename.sql
```

Créez un instantané de plusieurs bases de données:

```
mysqldump [options] --databases db_name1 db_name2 ... > filename.sql
mysqldump [options] --all-databases > filename.sql
```

Créez un instantané d'une ou plusieurs tables:

```
mysqldump [options] db_name table_name... > filename.sql
```

Créez un instantané *excluant* une ou plusieurs tables:

```
mysqldump [options] db_name --ignore-table=tbl1 --ignore-table=tbl2 ... > filename.sql
```

L'extension de fichier `.sql` est entièrement une question de style. Toute extension fonctionnerait.

## Spécifier le nom d'utilisateur et le mot de passe

```
> mysqldump -u username -p [other options]
Enter password:
```

Si vous devez spécifier le mot de passe sur la ligne de commande (par exemple dans un script), vous pouvez l'ajouter après l'option `-p` *sans* espace:

```
> mysqldump -u username -ppassword [other options]
```

Si votre mot de passe contient des espaces ou des caractères spéciaux, n'oubliez pas d'utiliser l'échappement en fonction de votre shell / système.

La forme étendue est facultativement:

```
> mysqldump --user=username --password=password [other options]
```

(L'explication spécifiant le mot de passe sur la ligne de commande n'est pas recommandée en raison de problèmes de sécurité.)

## Restauration d'une sauvegarde d'une base de données ou d'une table

```
mysql [options] db_name < filename.sql
```

Notez que:

- `db_name` doit être une base de données existante;
- votre utilisateur authentifié dispose de privilèges suffisants pour exécuter toutes les commandes de votre `filename.sql` ;
- L'extension de fichier `.sql` est entièrement une question de style. Toute extension fonctionnerait.
- Vous ne pouvez pas spécifier un nom de table à charger, même si vous pouvez en spécifier un pour le sauvegarder. Cela doit être fait avec `filename.sql` .

Sinon, lorsque vous utilisez l' **outil de ligne de commande MySQL** , vous pouvez restaurer (ou exécuter tout autre script) en utilisant la commande source:

```
source filename.sql
```

ou

```
\. filename.sql
```

## mysqldump depuis un serveur distant avec compression

Afin d'utiliser la compression sur le câble pour un transfert plus rapide, passez l'option `--compress` à `mysqldump`. Exemple:

```
mysqldump -h db.example.com -u username -p --compress dbname > dbname.sql
```

Important: Si vous ne souhaitez pas verrouiller la base de données *source*, vous devez également inclure `--lock-tables=false`. Mais vous ne pouvez pas obtenir une image de base de données cohérente en interne de cette façon.

Pour enregistrer également le fichier compressé, vous pouvez diriger vers `gzip`.

```
mysqldump -h db.example.com -u username -p --compress dbname | gzip --stdout > dbname.sql.gz
```

## restaurer un fichier mysqldump sans compresser

```
gunzip -c dbname.sql.gz | mysql dbname -u username -p
```

Remarque: `-c` signifie écrire la sortie sur stdout.

## Sauvegarde directe sur Amazon S3 avec compression

Si vous souhaitez effectuer une sauvegarde complète d'une installation MySQL volumineuse et que vous ne disposez pas d'un stockage local suffisant, vous pouvez le sauvegarder et le compresser directement dans un compartiment Amazon S3. Il est également conseillé de le faire sans avoir le mot de passe DB dans le cadre de la commande:

```
mysqldump -u root -p --host=localhost --opt --skip-lock-tables --single-transaction \  
--verbose --hex-blob --routines --triggers --all-databases | \  
gzip -9 | s3cmd put - s3://s3-bucket/db-server-name.sql.gz
```

Vous êtes invité à entrer le mot de passe, après quoi la sauvegarde démarre.

## Transfert de données d'un serveur MySQL vers un autre

Si vous devez copier une base de données d'un serveur à un autre, vous avez deux options:

### Option 1:

1. Stocker le fichier de vidage dans le serveur source



2. Copiez le fichier de vidage sur votre serveur de destination
3. Chargez le fichier de vidage dans votre serveur de destination

Sur le serveur source:

```
mysqldump [options] > dump.sql
```

Sur le serveur de destination, copiez le fichier de vidage et exécutez:

```
mysql [options] < dump.sql
```

## Option 2:

Si le serveur de destination peut se connecter au serveur hôte, vous pouvez utiliser un pipeline pour copier la base de données d'un serveur à l'autre:

Sur le serveur de destination

```
mysqldump [options to connect to the source server] | mysql [options]
```

De même, le script peut être exécuté sur le serveur source, en allant jusqu'à la destination. Dans les deux cas, il est susceptible d'être nettement plus rapide que l'option 1.

## Sauvegarde de la base de données avec des procédures stockées et des fonctions

Par défaut les procédures et fonctions stockées ou non générées par `mysqldump`, vous devrez ajouter le paramètre `--routines` (ou `-R`):

```
mysqldump -u username -p -R db_name > dump.sql
```

Lors de l'utilisation de `--routines` les `--routines` création et de modification ne sont pas conservés. Au lieu de cela, vous devez vider et recharger le contenu de `mysql.proc`.

Lire Sauvegarde avec mysqldump en ligne: <https://riptutorial.com/fr/mysql/topic/604/sauvegarde-avec-mysqldump>

---

# Chapitre 58: Sécurité via GRANTS

## Exemples

### Meilleur entraînement

Limiter la racine (et tout autre utilisateur privilégié par SUPER) à

```
GRANT ... TO root@localhost ...
```

Cela empêche l'accès des autres serveurs. Vous devez distribuer SUPER à très peu de gens et ils doivent être conscients de leurs responsabilités. L'application ne doit pas avoir SUPER.

Limitez les connexions de l'application à la seule base de données utilisée:

```
GRANT ... ON dbname.* ...
```

De cette façon, quelqu'un qui pirate le code de l'application ne peut pas obtenir dbname. Cela peut être affiné par l'une de ces deux méthodes:

```
GRANT SELECT ON dbname.* ... -- "read only"  
GRANT ... ON dbname.tblname ... -- "just one table"
```

Le readonly peut aussi avoir besoin de choses "sûres" comme

```
GRANT SELECT, CREATE TEMPORARY TABLE ON dbname.* ... -- "read only"
```

Comme vous le dites, il n'y a pas de sécurité absolue. Mon point ici est que vous pouvez faire quelques choses pour ralentir les pirates. (Même chose pour les honnêtes gens qui font des bêtises.)

Dans de rares cas, il se peut que l'application doive faire quelque chose de disponible uniquement pour la racine. Cela peut être fait via une "procédure stockée" qui a `SECURITY DEFINER` (et root la définit). Cela exposera uniquement ce que fait le SP, ce qui pourrait, par exemple, être une action particulière sur une table particulière.

### Hôte (de l'utilisateur @ hôte)

L'hôte peut être un nom d'hôte ou une adresse IP. En outre, il peut s'agir de jokers.

```
GRANT SELECT ON db.* TO sam@'my.domain.com' IDENTIFIED BY 'foo';
```

Exemples: Remarque: il faut généralement les citer

```
localhost -- the same machine as mysqld
```

```
'my.domain.com' -- a specific domain; this involves a lookup  
'11.22.33.44' -- a specific IP address  
'192.168.1.%' -- wild card for trailing part of IP address. (192.168.% and 10.% and 11.% are  
"internal" ip addresses.)
```

Utiliser `localhost` repose sur la sécurité du serveur. Pour de meilleures pratiques, `root` ne devrait être autorisé que via `localhost`. Dans certains cas, cela signifie la même chose: `0.0.0.1` et `:::1`.

Lire Sécurité via GRANTS en ligne: <https://riptutorial.com/fr/mysql/topic/5131/securite-via-grants>

# Chapitre 59: SÉLECTIONNER

## Introduction

`SELECT` permet de récupérer des lignes sélectionnées dans une ou plusieurs tables.

## Syntaxe

- `SELECT DISTINCT [expressions] FROM TableName [WHERE conditions];` // Simple Select
- `SELECT DISTINCT (a), b ...` est identique à `SELECT DISTINCT a, b ...`
- `SELECT [TOUS | DISTINCT | DISTINCTROW] [HIGH_PRIORITY] [STRAIGHT_JOIN] [SQL_SMALL_RESULT | SQL_BIG_RESULT] [SQL_BUFFER_RESULT] [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] expressions FROM tables [conditions WHERE] [expressions GROUP BY] [condition HAVING] [expression ORDER BY [ASC | DESC]] [LIMIT [offset_value] number_rows | LIMIT number_rows OFFSET offset_value] [PROCEDURE nom_procédure] [INTO [OUTFILE 'nom_fichier' options | DUMPFILE 'nom_fichier' | @ variable1, @ variable2, ... @variable_n] [POUR MISE À JOUR | VERROUILLAGE EN MODE SHARE];` // Syntaxe complète de sélection

## Remarques

Pour plus d'informations sur l' `SELECT` de `SELECT` , consultez [MySQL Docs](#) .

## Exemples

### SELECT par nom de colonne

```
CREATE TABLE stack(  
  id INT,  
  username VARCHAR(30) NOT NULL,  
  password VARCHAR(30) NOT NULL  
);  
  
INSERT INTO stack (`id`, `username`, `password`) VALUES (1, 'Foo', 'hiddenGem');  
INSERT INTO stack (`id`, `username`, `password`) VALUES (2, 'Baa', 'verySecret');
```

### Question

```
SELECT id FROM stack;
```

### Résultat

```
+-----+  
| id   |
```

```
+-----+
| 1 |
| 2 |
+-----+
```

## SÉLECTIONNEZ toutes les colonnes (\*)

### Question

```
SELECT * FROM stack;
```

### Résultat

```
+-----+-----+-----+
| id  | username | password |
+-----+-----+-----+
| 1   | admin   | admin   |
| 2   | stack   | stack   |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

Vous pouvez sélectionner toutes les colonnes d'une table dans une jointure en procédant comme suit:

```
SELECT stack.* FROM stack JOIN Overflow ON stack.id = Overflow.id;
```

**Meilleure pratique** Ne pas utiliser \* moins de déboguer ou d'extraire la ou les lignes dans des tableaux associatifs, faute de quoi les modifications du schéma (colonnes ADD / DROP / réorganiser) peuvent entraîner des erreurs d'application malveillantes. De plus, si vous donnez la liste des colonnes dont vous avez besoin dans votre jeu de résultats, le planificateur de requêtes de MySQL peut souvent optimiser la requête.

### Avantages:

1. Lorsque vous ajoutez / supprimez des colonnes, vous n'avez pas à apporter de modifications lorsque vous avez utilisé `SELECT *`
2. C'est plus court pour écrire
3. Vous voyez aussi les réponses, alors `SELECT *` -usage peut-il être justifié?

### Les inconvénients:

1. Vous retournez plus de données que nécessaire. Disons que vous ajoutez une colonne `VARBINARY` qui contient 200k par ligne. Vous n'avez besoin que de ces données à un seul endroit pour un seul enregistrement - en utilisant `SELECT *` vous pouvez finir par retourner 2 Mo par 10 lignes dont vous n'avez pas besoin
2. Expliquer les données utilisées
3. Spécifier des colonnes signifie que vous obtenez une erreur lorsqu'une colonne est supprimée
4. Le processeur de requêtes doit faire plus de travail - déterminer quelles colonnes existent

sur la table (merci @vinodadhikary)

5. Vous pouvez trouver où une colonne est utilisée plus facilement
6. Vous obtenez toutes les colonnes dans les jointures si vous utilisez `SELECT *`
7. Vous ne pouvez pas utiliser le référencement ordinal en toute sécurité (bien que l'utilisation de références ordinales pour les colonnes soit une mauvaise pratique en soi)
8. Dans les requêtes complexes avec des champs `TEXT`, la requête peut être ralentie par un traitement de table temporaire moins optimal

## SELECT avec WHERE

### Question

```
SELECT * FROM stack WHERE username = "admin" AND password = "admin";
```

### Résultat

```
+-----+-----+-----+
| id   | username | password |
+-----+-----+-----+
|    1 | admin   | admin   |
+-----+-----+-----+
1 row in set (0.00 sec)
```

## Requête avec un SELECT imbriqué dans la clause WHERE

La clause `WHERE` peut contenir toute instruction `SELECT` valide pour écrire des requêtes plus complexes. Ceci est une requête 'imbriquée'

### Question

Les requêtes imbriquées sont généralement utilisées pour renvoyer des valeurs atomiques uniques à partir de requêtes à des fins de comparaison.

```
SELECT title FROM books WHERE author_id = (SELECT id FROM authors WHERE last_name = 'Bar' AND first_name = 'Foo');
```

Sélectionne tous les noms d'utilisateur sans adresse e-mail

```
SELECT * FROM stack WHERE username IN (SELECT username FROM signups WHERE email IS NULL);
```

Clause de non-responsabilité: envisagez d'utiliser des [jointures](#) pour améliorer les performances lors de la comparaison d'un ensemble de résultats complet.

## SELECT avec LIKE (%)

```
CREATE TABLE stack
( id int AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(100) NOT NULL
);

INSERT stack(username) VALUES
('admin'),('k admin'),('adm'),('a adm b'),('b XadmY c'), ('adm now'), ('not here');
```

"adm" n'importe où:

```
SELECT * FROM stack WHERE username LIKE "%adm%";
+----+-----+
| id | username |
+----+-----+
| 1 | admin    |
| 2 | k admin  |
| 3 | adm      |
| 4 | a adm b  |
| 5 | b XadmY c|
| 6 | adm now  |
+----+-----+
```

Commence par "adm":

```
SELECT * FROM stack WHERE username LIKE "adm%";
+----+-----+
| id | username |
+----+-----+
| 1 | admin    |
| 3 | adm      |
| 6 | adm now  |
+----+-----+
```

Se termine par "adm":

```
SELECT * FROM stack WHERE username LIKE "%adm";
+----+-----+
| id | username |
+----+-----+
| 3 | adm      |
+----+-----+
```

Tout comme le caractère % d'une clause `LIKE` correspond à un nombre quelconque de caractères, le caractère `_` correspond qu'à un seul caractère. Par exemple,

```
SELECT * FROM stack WHERE username LIKE "adm_n";
+----+-----+
| id | username |
+----+-----+
| 1 | admin    |
+----+-----+
```

**Notes de performance** S'il existe un index sur le `username` d' `username` , alors

- `LIKE 'adm'` effectue la même chose que ``='adm'`

- `LIKE 'adm%` est une "range", similaire à `BETWEEN..AND..`. Elle peut faire bon usage d'un index sur la colonne.
- `LIKE '%adm'` (ou toute variante avec un caractère générique en tête) ne peut utiliser aucun index. Par conséquent, ce sera lent. Sur les tables comportant de nombreuses lignes, il est probable qu'elle soit si lente qu'elle est inutile.
- `RLIKE ( REGEXP )` a tendance à être plus lent que `LIKE`, mais a plus de capacités.
- Bien que MySQL offre l'indexation `FULLTEXT` sur de nombreux types de tables et de colonnes, ces index `FULLTEXT` ne sont pas utilisés pour répondre aux requêtes utilisant `LIKE`.

## SELECT avec Alias (AS)

Les alias SQL sont utilisés pour renommer temporairement une table ou une colonne. Ils sont généralement utilisés pour améliorer la lisibilité.

### Question

```
SELECT username AS val FROM stack;
SELECT username val FROM stack;
```

(Remarque: `AS` est syntaxiquement facultatif.)

### Résultat

```
+-----+
| val  |
+-----+
| admin|
| stack|
+-----+
2 rows in set (0.00 sec)
```

## SELECT avec une clause LIMIT

### Question:

```
SELECT *
FROM Customers
ORDER BY CustomerID
LIMIT 3;
```

### Résultat:

| N ° de client | Nom du client               | Nom du contact | Adresse                  | Ville      | Code postal | Pays      |
|---------------|-----------------------------|----------------|--------------------------|------------|-------------|-----------|
| 1             | Alfreds Futterkiste         | Maria Anders   | Obere Str. 57            | Berlin     | 12209       | Allemagne |
| 2             | Ana Trujillo Emparedados et | Ana Trujillo   | Avda. de la Constitución | Mexique DF | 05021       | Mexique   |



| helados |                         | 2222           |                |            |       |         |
|---------|-------------------------|----------------|----------------|------------|-------|---------|
| 3       | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | Mexique DF | 05023 | Mexique |

**Meilleure pratique** Utilisez toujours `ORDER BY` lorsque vous utilisez `LIMIT` ; sinon les lignes que vous obtiendrez seront imprévisibles.

### Question:

```
SELECT *
FROM Customers
ORDER BY CustomerID
LIMIT 2,1;
```

### Explication:

Lorsqu'une clause `LIMIT` contient deux nombres, elle est interprétée comme un `LIMIT offset, count`. Ainsi, dans cet exemple, la requête ignore deux enregistrements et en renvoie un.

### Résultat:

| N ° de client | Nom du client           | Nom du contact | Adresse        | Ville      | Code postal | Pays    |
|---------------|-------------------------|----------------|----------------|------------|-------------|---------|
| 3             | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | Mexique DF | 05023       | Mexique |

### Remarque:

Les valeurs des clauses `LIMIT` doivent être des constantes; ils peuvent ne pas être des valeurs de colonne.

## SELECT avec DISTINCT

La clause `DISTINCT` après `SELECT` élimine les lignes en double du jeu de résultats.

```
CREATE TABLE `car`
(
  `car_id` INT UNSIGNED NOT NULL PRIMARY KEY,
  `name` VARCHAR(20),
  `price` DECIMAL(8,2)
);

INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (1, 'Audi A1', '20000');
INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (2, 'Audi A1', '15000');
INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (3, 'Audi A2', '40000');
INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (4, 'Audi A2', '40000');

SELECT DISTINCT `name`, `price` FROM CAR;
+-----+-----+
| name  | price |
+-----+-----+
```

```
| Audi A1 | 20000.00 |
| Audi A1 | 15000.00 |
| Audi A2 | 40000.00 |
+-----+-----+
```

`DISTINCT` fonctionne sur toutes les colonnes pour fournir les résultats, pas les colonnes individuelles. Ce dernier est souvent une idée fautive des nouveaux développeurs SQL. En bref, c'est la distinction au niveau des lignes de l'ensemble de résultats qui compte, et non la distinction au niveau des colonnes. Pour visualiser cela, regardez "Audi A1" dans le jeu de résultats ci-dessus.

Pour les versions ultérieures de MySQL, `DISTINCT` a des implications sur son utilisation aux côtés de `ORDER BY`. Le paramètre pour `ONLY_FULL_GROUP_BY` entre en jeu, comme le montre la page du manuel MySQL suivante, intitulée [MySQL Handling of GROUP BY](#).

## SELECT avec LIKE (\_)

Un caractère `_` dans un motif de clause `LIKE` correspond à un seul caractère.

### Question

```
SELECT username FROM users WHERE users LIKE 'admin_';
```

### Résultat

```
+-----+
| username |
+-----+
| admin1   |
| admin2   |
| admin-   |
| adminA   |
+-----+
```

## SELECT avec CASE ou IF

### Question

```
SELECT st.name,
       st.percentage,
       CASE WHEN st.percentage >= 35 THEN 'Pass' ELSE 'Fail' END AS `Remark`
FROM student AS st ;
```

### Résultat

```
+-----+-----+-----+
| name   | percentage | Remark |
+-----+-----+-----+
| Isha   | 67         | Pass   |
| Rucha  | 28         | Fail   |
| Het    | 35         | Pass   |
```

```
| Ansh | 92 | Pass |
+-----+
```

## Ou avec IF

```
SELECT st.name,
       st.percentage,
       IF(st.percentage >= 35, 'Pass', 'Fail') AS `Remark`
FROM student AS st ;
```

## NB

```
IF(st.percentage >= 35, 'Pass', 'Fail')
```

Cela signifie que: si `st.percentage >= 35` est **TRUE**, alors retourne 'Pass' ELSE return 'Fail'

## SELECT avec entre

Vous pouvez utiliser la clause BETWEEN pour remplacer une combinaison de conditions "supérieures à égales ET inférieures à égales".

## Les données

```
+----+-----+
| id | username |
+----+-----+
| 1  | admin   |
| 2  | root    |
| 3  | toor    |
| 4  | mysql   |
| 5  | thanks  |
| 6  | java    |
+----+-----+
```

## Interroger avec des opérateurs

```
SELECT * FROM stack WHERE id >= 2 and id <= 5;
```

## Requête similaire avec BETWEEN

```
SELECT * FROM stack WHERE id BETWEEN 2 and 5;
```

## Résultat

```
+----+-----+
| id | username |
+----+-----+
| 2  | root     |
| 3  | toor     |
| 4  | mysql    |
+----+-----+
```

```
| 5 | thanks |
+----+-----+
4 rows in set (0.00 sec)
```

## Remarque

**BETWEEN** utilise `>=` et `<=`, non `>` et `<`.

## En utilisant pas entre

Si vous souhaitez utiliser le négatif, vous pouvez utiliser `NOT`. Par exemple :

```
SELECT * FROM stack WHERE id NOT BETWEEN 2 and 5;
```

## Résultat

```
+----+-----+
| id | username |
+----+-----+
| 1  | admin    |
| 6  | java     |
+----+-----+
2 rows in set (0.00 sec)
```

## Remarque

**NOT BETWEEN** utilise `>` et `<` et non `>=` et `<=`. C'est-à-dire, où `WHERE id NOT BETWEEN 2 and 5` est identique à `WHERE (id < 2 OR id > 5)`.

Si vous avez un index sur une colonne que vous utilisez dans une recherche `BETWEEN`, MySQL peut utiliser cet index pour une analyse de plage.

## SELECT avec plage de dates

```
SELECT ... WHERE dt >= '2017-02-01'
AND dt < '2017-02-01' + INTERVAL 1 MONTH
```

Bien sûr, cela pourrait être fait avec `BETWEEN` et l'inclusion de `23:59:59`. Mais, le modèle a ces avantages:

- Vous ne pré-calculez pas la date de fin (qui est souvent une longueur exacte depuis le début)
- Vous n'incluez pas les deux points de terminaison (comme le fait `BETWEEN`), ni tapez "23: 59: 59" pour l'éviter.
- Il fonctionne pour `DATE`, `TIMESTAMP`, `DATETIME` et même pour `DATETIME(6)` inclus dans la microseconde.
- Il prend en charge les jours bissextiles, en fin d'année, etc.
- Il est facile à index (donc est `BETWEEN`).

Lire **SÉLECTIONNER** en ligne: <https://riptutorial.com/fr/mysql/topic/3307/selectionner>

# Chapitre 60: SYNDICAT

## Syntaxe

- UNION DISTINCT - déduit après avoir combiné les SELECT
- UNION ALL - non dedup (plus rapide)
- UNION - la valeur par défaut est DISTINCT
- SELECT ... UNION SELECT ... - est OK, mais ambigu sur ORDER BY
- (SELECT ...) UNION (SELECT ...) ORDER BY ... - résout l'ambiguïté

## Remarques

UNION n'utilise pas plusieurs processeurs.

UNION toujours \* implique une table temporaire pour collecter les résultats. \* A partir de 5.7.3 / MariaDB 10.1, certaines formes d'UNION fournissent les résultats sans utiliser de table tmp (donc plus rapide).

## Exemples

### Combiner les instructions SELECT avec UNION

Vous pouvez combiner les résultats de deux requêtes à structure identique avec le mot-clé UNION .

Par exemple, si vous vouliez une liste de toutes les informations de contact de deux tables, `authors` et `editors` distincts, par exemple, vous pourriez utiliser le mot-clé UNION comme suit:

```
select name, email, phone_number
from authors

union

select name, email, phone_number
from editors
```

Utiliser l' `union` par elle-même supprimera les doublons. Si vous aviez besoin de conserver des doublons dans votre requête, vous pouvez utiliser le mot clé `ALL` comme suit: `UNION ALL` .

### COMMANDÉ PAR

Si vous devez trier les résultats d'un UNION, utilisez ce modèle:

```
( SELECT ... )
UNION
( SELECT ... )
ORDER BY
```

Sans les parenthèses, le dernier ORDER BY appartiendrait au dernier SELECT.

## Pagination via OFFSET

Lorsque vous ajoutez une LIMITE à une UNION, c'est le modèle à utiliser:

```
( SELECT ... ORDER BY x LIMIT 10 )  
UNION  
( SELECT ... ORDER BY x LIMIT 10 )  
ORDER BY x LIMIT 10
```

Comme vous ne pouvez pas prédire quel (s) SELECT (s) le "10" va provenir, vous devez obtenir 10 pour chacun, puis réduire la liste, en répétant à la fois ORDER BY et LIMIT .

Pour la 4ème page de 10 articles, ce modèle est nécessaire:

```
( SELECT ... ORDER BY x LIMIT 40 )  
UNION  
( SELECT ... ORDER BY x LIMIT 40 )  
ORDER BY x LIMIT 30, 10
```

Autrement dit, collectez 4 pages dans chaque SELECT , puis effectuez le OFFSET dans l' UNION .

## Combinaison de données avec différentes colonnes

```
SELECT name, caption as title, year, pages FROM books  
UNION  
SELECT name, title, year, 0 as pages FROM movies
```

Lors de la combinaison de 2 jeux d'enregistrements avec des colonnes différentes, émulez les éléments manquants avec des valeurs par défaut.

## UNION ALL et UNION

```
SELECT 1,22,44 UNION SELECT 2,33,55
```

| 信息  | 结果1 | 概况 | 状态 |
|-----|-----|----|----|
| 1   | 22  | 44 |    |
| ▶ 1 | 22  | 44 |    |
| 2   | 33  | 55 |    |

```
SELECT 1,22,44 UNION SELECT 2,33,55 UNION SELECT 2,33,55
```

**Le résultat est le même que ci-dessus.**

utiliser UNION ALL

quand

SELECT 1,22,44 UNION SELECT 2,33,55 UNION ALL SELECT 2,33,55

| 信息  | 结果1 | 概况 | 状态 |
|-----|-----|----|----|
| 1   | 22  | 44 |    |
| ▶ 1 | 22  | 44 |    |
| 2   | 33  | 55 |    |
| 2   | 33  | 55 |    |

## Combinaison et fusion de données sur différentes tables MySQL avec les mêmes colonnes en lignes uniques et requête en cours d'exécution

Cet **UNION ALL** combine des données provenant de plusieurs tables et sert d'alias de nom de table à utiliser pour vos requêtes:

```
SELECT YEAR(date_time_column), MONTH(date_time_column), MIN(DATE(date_time_column)),
MAX(DATE(date_time_column)), COUNT(DISTINCT (ip)), COUNT(ip), (COUNT(ip) / COUNT(DISTINCT
(ip))) AS Ratio
FROM (
    (SELECT date_time_column, ip FROM server_log_1 WHERE state = 'action' AND log_id = 150)
UNION ALL
    (SELECT date_time_column, ip FROM server_log_2 WHERE state = 'action' AND log_id = 150)
UNION ALL
    (SELECT date_time_column, ip FROM server_log_3 WHERE state = 'action' AND log_id = 150)
UNION ALL
    (SELECT date_time_column, ip FROM server_log WHERE state = 'action' AND log_id = 150)
) AS table_all
GROUP BY YEAR(date_time_column), MONTH(date_time_column);
```

Lire SYNDICAT en ligne: <https://riptutorial.com/fr/mysql/topic/3847/syndicat>

# Chapitre 61: Table de chute

## Syntaxe

- DROP TABLE nom\_table;
- DROP TABLE IF EXISTS nom\_table; - pour éviter les erreurs embêtantes dans le script automatisé
- DROP TABLE t1, t2, t3; - DROP multiple tables
- DROP TEMPORARY TABLE t; - DROP une table de CREATE TEMPORARY TABLE ...

## Paramètres

| Paramètres | Détails  |
|------------|--|
| TEMPORAIRE | Optionnel. Il spécifie que seules les tables temporaires doivent être supprimées par l'instruction DROP TABLE. |
| SI EXISTE  | Optionnel. Si spécifié, l'instruction DROP TABLE ne générera pas d'erreur si l'une des tables n'existe pas.    |

## Exemples

### Table de chute

Drop Table est utilisé pour supprimer la table de la base de données.

### Création de tableau:

Création d'une table nommée tbl, puis suppression de la table créée

```
CREATE TABLE tbl(  
  id INT NOT NULL AUTO_INCREMENT,  
  title VARCHAR(100) NOT NULL,  
  author VARCHAR(40) NOT NULL,  
  submission_date DATE,  
  PRIMARY KEY (id)  
);
```

### Table de chute:

```
DROP TABLE tbl;
```

### NOTEZ S'IL VOUS PLAÎT

La table de suppression supprime complètement la table de la base de données et



toutes ses informations, et elle ne sera pas récupérée.

## Déposer des tables de la base de données

```
DROP TABLE Database.nom_table
```

Lire Table de chute en ligne: <https://riptutorial.com/fr/mysql/topic/4123/table-de-chute>

# Chapitre 62: Table de mappage plusieurs-à-plusieurs

## Remarques

- Absence d'un identifiant `AUTO_INCREMENT` pour cette table - La PK donnée est la PK «naturelle»; il n'y a pas de bonne raison pour un substitut.
- `MEDIUMINT` - Ceci est un rappel que toutes les `INTs` doivent être aussi petites que sûres (plus petites  $\Rightarrow$  plus rapides). Bien entendu, la déclaration ici doit correspondre à la définition du tableau lié.
- `UNSIGNED` - Presque toutes les `INT` peuvent aussi être déclarées non négatives
- `NOT NULL` - C'est vrai, n'est-ce pas?
- `InnoDB` - Plus efficace que `MyISAM` en raison de la façon dont la `PRIMARY KEY` est regroupée avec les données dans `InnoDB`.
- `INDEX(y_id, x_id)` - La `PRIMARY KEY` rend efficace pour aller dans une direction; le rend l'autre direction efficace. Pas besoin de dire `UNIQUE`; ce serait un effort supplémentaire sur les `INSERTs`.
- Dans l'index secondaire, dire juste `INDEX(y_id)` fonctionnerait car cela impliquerait d'inclure `x_id`. Mais je préférerais qu'il soit plus évident que j'espère un indice «couvrant».

Vous voudrez *peut-être* ajouter plus de colonnes à la table; c'est rare. Les colonnes supplémentaires peuvent fournir des informations sur la *relation* que représente la table.

Vous souhaitez *peut-être* ajouter des contraintes `FOREIGN KEY`.

## Exemples

### Schéma typique

```
CREATE TABLE XtoY (  
  # No surrogate id for this table  
  x_id MEDIUMINT UNSIGNED NOT NULL,    -- For JOINing to one table  
  y_id MEDIUMINT UNSIGNED NOT NULL,    -- For JOINing to the other table  
  # Include other fields specific to the 'relation'  
  PRIMARY KEY(x_id, y_id),             -- When starting with X  
  INDEX      (y_id, x_id)               -- When starting with Y  
) ENGINE=InnoDB;
```

(Voir les remarques ci-dessous pour la justification.)

Lire Table de mappage plusieurs-à-plusieurs en ligne:

<https://riptutorial.com/fr/mysql/topic/4857/table-de-mappage-plusieurs-a-plusieurs>

# Chapitre 63: Tableau dynamique de pivotement à l'aide de l'instruction préparée

## Exemples

### Dé-pivoter un ensemble dynamique de colonnes en fonction de la condition

L'exemple suivant est très utile lorsque vous tentez de convertir des données de transaction en données non pivotées pour des raisons de BI / de génération de rapports, où les dimensions à désassembler peuvent avoir un ensemble dynamique de colonnes.

Pour notre exemple, nous supposons que la table de données brutes contient des données d'évaluation d'employés sous la forme de questions marquées.

La table de données brutes est la suivante:

```
create table rawdata
(
  PersonId VARCHAR(255)
,Question1Id INT(11)
,Question2Id INT(11)
,Question3Id INT(11)
)
```

La table rawdata est une table temporaire faisant partie de la procédure ETL et peut comporter un nombre variable de questions. Le but est d'utiliser la même procédure de non-pivotement pour un nombre arbitraire de Questions, à savoir les colonnes qui vont être non pivotées.

Voici un exemple de jouet de la table rawdata:

|  | PersonId  | Question1Id | Question2Id | Question3Id |
|--|-----------|-------------|-------------|-------------|
|  | Giannaros | 1           | 3           | 1           |
|  | Patra     | 2           | 4           | 3           |

La méthode statique connue pour décomposer les données dans MYSQL consiste à utiliser UNION ALL:

```
create table unpivoteddata
(
  PersonId VARCHAR(255)
,QuestionId VARCHAR(255)
,QuestionValue INT(11)
);

INSERT INTO unpivoteddata SELECT PersonId, 'Question1Id' col, Question1Id
```

```

FROM rawdata
UNION ALL
SELECT PersonId, 'Question2Id' col, Question2Id
FROM rawdata
UNION ALL
SELECT PersonId, 'Question3Id' col, Question3Id
FROM rawdata;

```

Dans notre cas, nous voulons définir un moyen de décomposer un nombre arbitraire de colonnes QuestionId. Pour cela, nous devons exécuter une instruction préparée qui est une sélection dynamique des colonnes souhaitées. Afin de pouvoir choisir quelles colonnes doivent être non pivotées, nous utiliserons une instruction GROUP\_CONCAT et nous choisirons les colonnes pour lesquelles le type de données est défini sur 'int'. Dans GROUP\_CONCAT, nous incluons également tous les éléments supplémentaires de notre instruction SELECT à exécuter.

```

set @temp2 = null;

SELECT GROUP_CONCAT(' SELECT ', 'PersonId', ',', ' ', COLUMN_NAME, ' ', ' col
', ',', ' ', COLUMN_NAME, ' FROM rawdata' separator ' UNION ALL' ) FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'rawdata' AND DATA_TYPE = 'Int' INTO @temp2;

select @temp2;

```

À une autre occasion, nous aurions pu choisir des colonnes indiquant que le nom de la colonne correspond à un motif, par exemple au lieu de

```
DATA_TYPE = 'Int'
```

utilisation

```
COLUMN_NAME LIKE 'Question%'
```

ou quelque chose qui peut être contrôlé par la phase ETL.

La déclaration préparée est finalisée comme suit:

```

set @temp3 = null;

select concat('INSERT INTO unpivoteddata', @temp2) INTO @temp3;

select @temp3;

prepare stmt FROM @temp3;
execute stmt;
deallocate prepare stmt;

```

La table unpivoteddata est la suivante:

```
SELECT * FROM unpivoteddata
```

| PersonId  | QuestionId  | QuestionValue |
|-----------|-------------|---------------|
| Giannaros | Question1Id | 1             |
| Patra     | Question1Id | 2             |
| Giannaros | Question2Id | 3             |
| Patra     | Question2Id | 4             |
| Giannaros | Question3Id | 1             |
| Patra     | Question3Id | 3             |

La sélection de colonnes en fonction d'une condition et la création d'une instruction préparée constituent un moyen efficace de désynchroniser dynamiquement les données.

Lire [Tableau dynamique de pivotement à l'aide de l'instruction préparée en ligne](https://riptutorial.com/fr/mysql/topic/6491/tableau-dynamique-de-pivotement-a-l-aide-de-l-instruction-prepree):  
<https://riptutorial.com/fr/mysql/topic/6491/tableau-dynamique-de-pivotement-a-l-aide-de-l-instruction-prepree>

# Chapitre 64: Tables temporaires

## Exemples

### Créer une table temporaire

Les tables temporaires pourraient être très utiles pour conserver des données temporaires. L'option des tables temporaires est disponible dans MySQL version 3.23 et supérieure.

La table temporaire sera automatiquement détruite à la fin de la session ou à la fermeture de la connexion. L'utilisateur peut également supprimer une table temporaire.

Le même nom de table temporaire peut être utilisé dans plusieurs connexions simultanément, car la table temporaire est uniquement disponible et accessible par le client qui crée cette table.

La table temporaire peut être créée dans les types suivants

```
--->Basic temporary table creation
CREATE TEMPORARY TABLE tempTable1(
    id INT NOT NULL AUTO_INCREMENT,
    title VARCHAR(100) NOT NULL,
    PRIMARY KEY ( id )
);

--->Temporary table creation from select query
CREATE TEMPORARY TABLE tempTable1
SELECT ColumnName1,ColumnName2,... FROM table1;
```

Vous pouvez ajouter des index lorsque vous créez la table:

```
CREATE TEMPORARY TABLE tempTable1
( PRIMARY KEY(ColumnName2) )
SELECT ColumnName1,ColumnName2,... FROM table1;
```

`IF NOT EXISTS` mot clé peut être utilisé comme mentionné ci-dessous pour éviter l'erreur «*table existe déjà*». Mais dans ce cas, la table ne sera pas créée si le nom de la table que vous utilisez existe déjà dans votre session en cours.

```
CREATE TEMPORARY TABLE IF NOT EXISTS tempTable1
SELECT ColumnName1,ColumnName2,... FROM table1;
```

### Drop Table temporaire

Drop Table temporaire est utilisée pour supprimer la table temporaire que vous avez créée dans votre session en cours.

```
DROP TEMPORARY TABLE tempTable1
```

```
DROP TEMPORARY TABLE IF EXISTS tempTable1
```

Utilisez `IF EXISTS` pour éviter une erreur sur les tables qui pourraient ne pas exister

Lire Tables temporaires en ligne: <https://riptutorial.com/fr/mysql/topic/5757/tables-temporaires>

---

# Chapitre 65: Temps avec précision de la seconde

## Remarques

Vous devez être à la version 5.6.4 ou ultérieure de MySQL pour déclarer des colonnes avec des types de données de fraction de seconde.

Par exemple, `DATETIME(3)` vous donnera une résolution en millisecondes dans vos horodatages et `TIMESTAMP(6)` vous donnera une résolution en microsecondes sur un horodatage de style \* nix.

Lisez ceci: <http://dev.mysql.com/doc/refman/5.7/fr/fractional-seconds.html>

`NOW(3)` vous donnera l'heure actuelle du système d'exploitation de votre serveur MySQL avec une précision en milliseconde.

(Notez que l'arithmétique fractionnaire interne de MySQL, comme \* 0.001, est toujours traitée comme une virgule flottante à double précision IEEE754. Il est donc peu probable que vous perdiez en précision avant que le soleil ne devienne une étoile naine blanche.)

## Exemples

### Obtenez l'heure actuelle avec une précision en milliseconde

```
SELECT NOW(3)
```

fait le tour

### Obtenez l'heure actuelle sous une forme qui ressemble à un horodatage Javascript.

Les horodatages Javascript sont basés sur le type de données vénérable UNIX `time_t` et indiquent le nombre de millisecondes `1970-01-01 00:00:00` depuis le `1970-01-01 00:00:00` UTC.

Cette expression obtient l'heure actuelle en tant qu'entier d'horodatage Javascript. (Il le fait correctement quel que soit le paramètre `time_zone` actuel.)

```
ROUND (UNIX_TIMESTAMP (NOW(3)) * 1000.0, 0)
```

Si vous avez des valeurs `TIMESTAMP` stockées dans une colonne, vous pouvez les récupérer en tant qu'horodatage Javascript entier à l'aide de la fonction `UNIX_TIMESTAMP()`.

```
SELECT ROUND (UNIX_TIMESTAMP (column) * 1000.0, 0)
```



Si votre colonne contient des colonnes `DATETIME` et que vous les récupérez comme horodatage Javascript, ces horodatages seront compensés par le décalage horaire du fuseau horaire dans lequel ils sont stockés.

## Créez un tableau avec des colonnes pour stocker le temps inférieur à la seconde.

```
CREATE TABLE times (  
    dt DATETIME(3),  
    ts TIMESTAMP(3)  
);
```

créer une table avec des champs date / heure de précision en millisecondes.

```
INSERT INTO times VALUES (NOW(3), NOW(3));
```

Insère une ligne contenant les valeurs `NOW()` avec une précision en millisecondes dans la table.

```
INSERT INTO times VALUES ('2015-01-01 16:34:00.123', '2015-01-01 16:34:00.128');
```

insère des valeurs de précision spécifiques à la milliseconde.

**Notez** que vous devez utiliser `NOW(3)` plutôt que `NOW()` si vous utilisez cette fonction pour insérer des valeurs de temps de haute précision.

## Convertissez une valeur de date / heure en millisecondes de précision en texte.

`%f` est le spécificateur de format de précision fractionnaire pour [la fonction DATE\\_FORMAT \(\)](#) .

```
SELECT DATE_FORMAT(NOW(3), '%Y-%m-%d %H:%i:%s.%f')
```

affiche une valeur comme `2016-11-19 09:52:53.248000` avec microsecondes fractionnaires. Parce que nous avons utilisé `NOW(3)` , les trois derniers chiffres de la fraction sont 0.

## Stocker un horodatage Javascript dans une colonne `TIMESTAMP`

Si vous avez une valeur d'horodatage Javascript, par exemple `1478960868932` , vous pouvez la convertir en une valeur temporelle fractionnelle MySQL comme ceci:

```
FROM_UNIXTIME(1478960868932 * 0.001)
```

Il est simple d'utiliser ce type d'expression pour stocker votre horodatage Javascript dans une table MySQL. Faites ceci:

```
INSERT INTO table (col) VALUES (FROM_UNIXTIME(1478960868932 * 0.001))
```

(Évidemment, vous voudrez insérer d'autres colonnes.)

Lire Temps avec précision de la seconde en ligne:

<https://riptutorial.com/fr/mysql/topic/7850/temps-avec-precision-de-la-seconde>

# Chapitre 66: Traiter des données rares ou manquantes

## Exemples

### Utilisation de colonnes contenant des valeurs NULL

Dans MySQL et d'autres dialectes SQL, les valeurs `NULL` ont des propriétés spéciales.

Considérez le tableau suivant contenant les candidats, les entreprises pour lesquelles ils ont travaillé et la date à laquelle ils ont quitté l'entreprise. `NULL` indique qu'un candidat travaille toujours dans l'entreprise:

```
CREATE TABLE example
(`applicant_id` INT, `company_name` VARCHAR(255), `end_date` DATE);
```

| applicant_id | company_name    | end_date   |
|--------------|-----------------|------------|
| 1            | Google          | NULL       |
| 1            | Initech         | 2013-01-31 |
| 2            | Woodworking.com | 2016-08-25 |
| 2            | NY Times        | 2013-11-10 |
| 3            | NFL.com         | 2014-04-13 |

Votre tâche consiste à composer une requête qui renvoie toutes les lignes après le `2016-01-01`, y compris les employés travaillant toujours dans une entreprise (ceux dont la date de fin est `NULL`). Cette instruction `select`:

```
SELECT * FROM example WHERE end_date > '2016-01-01';
```

ne parvient pas à inclure des lignes avec des valeurs `NULL` :

| applicant_id | company_name    | end_date   |
|--------------|-----------------|------------|
| 2            | Woodworking.com | 2016-08-25 |

Selon la [documentation MySQL](#), les comparaisons utilisant les opérateurs arithmétiques `<`, `>`, `=` et `<>` renvoient elles-mêmes `NULL` au lieu de booléen `TRUE` ou `FALSE`. Ainsi, une ligne avec un `end_date` `NULL` n'est ni supérieure à `2016-01-01` ni inférieure à `2016-01-01`.

Cela peut être résolu en utilisant les mots-clés `IS NULL`:

```
SELECT * FROM example WHERE end_date > '2016-01-01' OR end_date IS NULL;
```

```

+-----+-----+-----+
| applicant_id | company_name | end_date |
+-----+-----+-----+
|           1 | Google       | NULL     |
|           2 | Woodworking.com | 2016-08-25 |
+-----+-----+-----+

```

Travailler avec des valeurs NULL devient plus complexe lorsque la tâche implique des fonctions d'agrégation telles que `MAX()` et une clause `GROUP BY`. Si votre tâche consistait à sélectionner la date d'emploi la plus récente pour chaque `demandeur_id`, la requête suivante semblerait être une première tentative logique:

```
SELECT applicant_id, MAX(end_date) FROM example GROUP BY applicant_id;
```

```

+-----+-----+-----+
| applicant_id | MAX(end_date) |
+-----+-----+-----+
|           1 | 2013-01-31   |
|           2 | 2016-08-25   |
|           3 | 2014-04-13   |
+-----+-----+-----+

```

Cependant, sachant que NULL indique qu'un candidat est toujours employé dans une entreprise, la première ligne du résultat est inexacte. L'utilisation de `CASE WHEN` fournit une solution de contournement pour le problème NULL :

```

SELECT
  applicant_id,
  CASE WHEN MAX(end_date is null) = 1 THEN 'present' ELSE MAX(end_date) END
  max_date
FROM example
GROUP BY applicant_id;

```

```

+-----+-----+-----+
| applicant_id | max_date      |
+-----+-----+-----+
|           1 | present       |
|           2 | 2016-08-25   |
|           3 | 2014-04-13   |
+-----+-----+-----+

```

Ce résultat peut être renvoyé à l'`example` tableau d'origine pour déterminer la société à laquelle un candidat a travaillé en dernier lieu:

```

SELECT
  data.applicant_id,
  data.company_name,
  data.max_date
FROM (
  SELECT
    *,
    CASE WHEN end_date is null THEN 'present' ELSE end_date END max_date
  FROM example
) data
INNER JOIN (

```

```

SELECT
  applicant_id,
  CASE WHEN MAX(end_date is null) = 1 THEN 'present' ELSE MAX(end_date) END max_date
FROM
  example
GROUP BY applicant_id
) j
ON data.applicant_id = j.applicant_id AND data.max_date = j.max_date;

```

```

+-----+-----+-----+
| applicant_id | company_name      | max_date  |
+-----+-----+-----+
|          1 | Google            | present   |
|          2 | Woodworking.com   | 2016-08-25 |
|          3 | NFL.com           | 2014-04-13 |
+-----+-----+-----+

```

Ce ne sont que quelques exemples de travail avec des valeurs `NULL` dans MySQL.

Lire [Traiter des données rares ou manquantes en ligne](https://riptutorial.com/fr/mysql/topic/5866/traiter-des-donnees-rares-ou-manquantes):

<https://riptutorial.com/fr/mysql/topic/5866/traiter-des-donnees-rares-ou-manquantes>

---

# Chapitre 67: Transaction

## Exemples

### Commencer la transaction

Une transaction est un groupe séquentiel d'instructions SQL telles que select, insert, update ou delete, qui se déroule en une seule unité de travail.

En d'autres termes, une transaction ne sera jamais complète à moins que chaque opération individuelle du groupe ne soit réussie. Si une opération dans la transaction échoue, la transaction entière échouera.

La transaction bancaire sera le meilleur exemple pour expliquer cela. Envisagez un transfert entre deux comptes. Pour ce faire, vous devez écrire des instructions SQL qui effectuent les opérations suivantes:

1. Vérifier la disponibilité du montant demandé dans le premier compte
2. Déduire le montant demandé du premier compte
3. Déposez-le dans un deuxième compte

Si quelqu'un de ce processus échoue, le tout devrait être rétabli à son état précédent.

### **ACID: Propriétés des transactions**

Les transactions ont les quatre propriétés standard suivantes

- **Atomicité:** s'assure que toutes les opérations dans l'unité de travail sont terminées avec succès; sinon, la transaction est abandonnée au moment de la défaillance et les opérations précédentes sont restaurées dans leur état antérieur.
- **Cohérence:** garantit que la base de données modifie correctement les états lors d'une transaction correctement validée.
- **Isolation:** permet aux transactions de fonctionner indépendamment les unes des autres et de manière transparente.
- **Durabilité:** garantit que le résultat ou l'effet d'une transaction validée persiste en cas de défaillance du système.

Les transactions commencent par l'instruction `START TRANSACTION` ou `BEGIN WORK` et se terminent par une instruction `COMMIT` ou `ROLLBACK`. Les commandes SQL entre les instructions de début et de fin constituent le gros de la transaction.

```
START TRANSACTION;
SET @transAmt = '500';
SELECT @availableAmt:=ledgerAmt FROM accTable WHERE customerId=1 FOR UPDATE;
UPDATE accTable SET ledgerAmt=ledgerAmt-@transAmt WHERE customerId=1;
UPDATE accTable SET ledgerAmt=ledgerAmt+@transAmt WHERE customerId=2;
COMMIT;
```

Avec `START TRANSACTION` , la validation automatique reste désactivée jusqu'à la fin de la transaction avec `COMMIT` ou `ROLLBACK` . Le mode autocommit revient alors à son état précédent.

Le `FOR UPDATE` indique (et verrouille) la ou les lignes pour la durée de la transaction.

Bien que la transaction ne soit pas validée, cette transaction ne sera pas disponible pour les autres utilisateurs.

## Procédures générales impliquées dans la transaction

- Commencez la transaction en émettant la commande SQL `BEGIN WORK` ou `START TRANSACTION` .
- Exécutez toutes vos instructions SQL.
- Vérifiez si tout est exécuté selon vos besoins.
- Si oui, `COMMIT` commande `COMMIT` , sinon, `ROLLBACK` une commande `ROLLBACK` pour revenir à l'état précédent.
- Vérifiez les erreurs même après `COMMIT` si vous utilisez, ou pourriez éventuellement utiliser, Galera / PXC.

## COMMIT, ROLLBACK et AUTOCOMMIT

### AUTOCOMMIT

MySQL valide automatiquement les instructions qui ne font pas partie d'une transaction. Les résultats de toute `UPDATE` , `DELETE` ou `INSERT` non précédée d'un `BEGIN` ou d'un `START TRANSACTION` seront immédiatement visibles pour toutes les connexions.

La variable `AUTOCOMMIT` est définie sur `true` par défaut. Cela peut être changé de la manière suivante,

```
--->To make autocommit false
SET AUTOCOMMIT=false;
--or
SET AUTOCOMMIT=0;

--->To make autocommit true
SET AUTOCOMMIT=true;
--or
SET AUTOCOMMIT=1;
```

Pour afficher le statut `AUTOCOMMIT`

```
SELECT @@autocommit;
```

### COMMETTRE

Si `AUTOCOMMIT` défini sur `false` et que la transaction n'est pas validée, les modifications ne seront visibles que pour la connexion en cours.

Une fois que l'instruction `COMMIT` validé les modifications apportées à la table, le résultat sera visible pour toutes les connexions.

Nous considérons deux connexions pour expliquer cela

### Connexion 1

```
--->Before making autocommit false one row added in a new table
mysql> INSERT INTO testTable VALUES (1);

--->Making autocommit = false
mysql> SET autocommit=0;

mysql> INSERT INTO testTable VALUES (2), (3);
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|  1  |
|  2  |
|  3  |
+-----+
```

### Connexion 2

```
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|  1  |
+-----+
---> Row inserted before autocommit=false only visible here
```

### Connexion 1

```
mysql> COMMIT;
--->Now COMMIT is executed in connection 1
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|  1  |
|  2  |
|  3  |
+-----+
```

### Connexion 2

```
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|  1  |
|  2  |
|  3  |
+-----+
--->Now all the three rows are visible here
```

## ROLLBACK



Si quelque chose a mal tourné dans l'exécution de votre requête, `ROLLBACK` utilisé pour annuler les modifications. Voir l'explication ci-dessous

```
--->Before making autocommit false one row added in a new table
mysql> INSERT INTO testTable VALUES (1);

--->Making autocommit = false
mysql> SET autocommit=0;

mysql> INSERT INTO testTable VALUES (2), (3);
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|  1  |
|  2  |
|  3  |
+-----+
```

Maintenant, nous exécutons `ROLLBACK`

```
--->Rollback executed now
mysql> ROLLBACK;

mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|  1  |
+-----+
--->Rollback removed all rows which all are not committed
```

Une fois que `COMMIT` est exécuté, alors `ROLLBACK` ne causera rien

```
mysql> INSERT INTO testTable VALUES (2), (3);
mysql> SELECT * FROM testTable;
mysql> COMMIT;
+-----+
| tId |
+-----+
|  1  |
|  2  |
|  3  |
+-----+

--->Rollback executed now
mysql> ROLLBACK;

mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|  1  |
|  2  |
|  3  |
+-----+
--->Rollback not removed any rows
```

Si `AUTOCOMMIT` est défini sur `true`, `COMMIT` et `ROLLBACK` sont inutiles

## Transaction à l'aide du pilote JDBC

La transaction utilisant le pilote JDBC est utilisée pour contrôler quand et comment une transaction doit être validée et annulée. La connexion au serveur MySQL est créée à l'aide du pilote JDBC

[Le pilote JDBC pour MySQL](#) peut être téléchargé ici

Commençons par obtenir une connexion à la base de données à l'aide du pilote JDBC

```
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection(DB_CONNECTION_URL,DB_USER,USER_PASSWORD);
-->Example for connection url "jdbc:mysql://localhost:3306/testDB";
```

**Jeux de caractères** : indique le jeu de caractères que le client utilisera pour envoyer des instructions SQL au serveur. Il spécifie également le jeu de caractères que le serveur doit utiliser pour renvoyer les résultats au client.

Cela doit être mentionné lors de la création de la connexion au serveur. Donc, la chaîne de connexion devrait être comme,

```
jdbc:mysql://localhost:3306/testDB?useUnicode=true&characterEncoding=utf8
```

Voir ceci pour plus de détails sur les [jeux de caractères et les assemblages](#)

Lorsque vous ouvrez une connexion, le mode `AUTOCOMMIT` est défini sur `true` par défaut, il doit être modifié sur `false` pour démarrer la transaction.

```
con.setAutoCommit(false);
```

Vous devez toujours appeler la méthode `setAutoCommit()` juste après avoir ouvert une connexion.

Sinon, utilisez `START TRANSACTION` ou `BEGIN WORK` pour lancer une nouvelle transaction. En utilisant `START TRANSACTION` ou `BEGIN WORK`, pas besoin de changer `AUTOCOMMIT` `false`. Cela sera automatiquement désactivé.

Maintenant, vous pouvez commencer la transaction. Voir un exemple de transaction JDBC complet ci-dessous.

```
package jdbcTest;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class accTrans {
```

```

public static void doTransfer(double transAmount,int customerIdFrom,int customerIdTo) {

    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        String DB_CONNECTION_URL =
"jdbc:mysql://localhost:3306/testDB?useUnicode=true&characterEncoding=utf8";

        Class.forName("com.mysql.jdbc.Driver");
        con = DriverManager.getConnection(DB_CONNECTION_URL,DB_USER,USER_PASSWORD);

        --->set auto commit to false
        con.setAutoCommit(false);
        ---> or use con.START TRANSACTION / con.BEGIN WORK

        --->Start SQL Statements for transaction
        --->Checking availability of amount
        double availableAmt    = 0;
        pstmt = con.prepareStatement("SELECT ledgerAmt FROM accTable WHERE customerId=?
FOR UPDATE");
        pstmt.setInt(1, customerIdFrom);
        rs = pstmt.executeQuery();
        if(rs.next())
            availableAmt    = rs.getDouble(1);

        if(availableAmt >= transAmount)
        {
            ---> Do Transfer
            ---> taking amount from cutomerIdFrom
            pstmt = con.prepareStatement("UPDATE accTable SET ledgerAmt=ledgerAmt-? WHERE
customerId=?");
            pstmt.setDouble(1, transAmount);
            pstmt.setInt(2, customerIdFrom);
            pstmt.executeUpdate();

            ---> depositing amount in cutomerIdTo
            pstmt = con.prepareStatement("UPDATE accTable SET ledgerAmt=ledgerAmt+? WHERE
customerId=?");
            pstmt.setDouble(1, transAmount);
            pstmt.setInt(2, customerIdTo);
            pstmt.executeUpdate();

            con.commit();
        }
        --->If you performed any insert,update or delete operations before
        ----> this availability check, then include this else part
        /*else { --->Rollback the transaction if availability is less than required
            con.rollback();
        }*/

    } catch (SQLException ex) {
        ---> Rollback the transaction in case of any error
        con.rollback();
    } finally {
        try {
            if(rs != null) rs.close();
            if(pstmt != null) pstmt.close();
            if(con != null) con.close();
        }
    }
}

```

```
    }  
}  
  
public static void main(String[] args) {  
    doTransfer(500, 1020, 1021);  
    -->doTransfer(transAmount, customerIdFrom, customerIdTo);  
}  
}
```

La transaction JDBC vérifie que toutes les instructions SQL d'un bloc de transaction ont été exécutées avec succès, si l'une des instructions SQL du bloc de transaction a échoué, annulez et annulez tout dans le bloc de transaction.

Lire Transaction en ligne: <https://riptutorial.com/fr/mysql/topic/5771/transaction>

---

# Chapitre 68: TRIGGERS

## Syntaxe

- `CREATE [DEFINER = {utilisateur | CURRENT_USER}] TRIGGER trigger_name trigger_time trigger_event ON nom_de_table POUR CHAQUE ROW [trigger_order] trigger_body`
- `trigger_time`: {AVANT | APRÈS }
- `trigger_event`: {INSERT | Mise à jour | EFFACER }
- `trigger_order`: {suit | PRECEDES} other\_trigger\_name

## Remarques

Deux points doivent attirer votre attention si vous utilisez déjà des déclencheurs sur d'autres bases de données:

---

## POUR CHAQUE RANG

`FOR EACH ROW` est une partie obligatoire de la syntaxe

Vous ne pouvez pas déclencher une *instruction* (une fois par requête) comme Oracle. C'est plus un problème de performance qu'une véritable fonctionnalité manquante

---

## Créer ou remplacer un déclencheur

Le `CREATE OR REPLACE` n'est pas supporté par MySQL

MySQL n'autorise pas cette syntaxe, vous devez plutôt utiliser les éléments suivants:

```
DELIMITER $$

DROP TRIGGER IF EXISTS myTrigger;
$$
CREATE TRIGGER myTrigger
-- ...

$$
DELIMITER ;
```

Attention, ce n'est **pas une transaction atomique** :

- vous perdrez l'ancien déclencheur si le `CREATE` échoue
- sur une charge lourde, d'autres opérations peuvent se `LOCK TABLES myTable WRITE`; entre le `DROP` et le `CREATE`, utilisez un `LOCK TABLES myTable WRITE`; tout d'abord pour éviter les incohérences de données et `UNLOCK TABLES`; après le `CREATE` pour libérer la table

# Exemples

## Déclencheur de base

### Créer une table

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));  
Query OK, 0 rows affected (0.03 sec)
```

### Créer un déclencheur

```
mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account  
-> FOR EACH ROW SET @sum = @sum + NEW.amount;  
Query OK, 0 rows affected (0.06 sec)
```

L'instruction `CREATE TRIGGER` crée un déclencheur nommé `ins_sum` associé à la table de compte. Il comprend également des clauses spécifiant le temps d'action du déclencheur, l'événement déclencheur et les actions à entreprendre lorsque le déclencheur est activé.

### Insérer une valeur

Pour utiliser le déclencheur, définissez la variable d'accumulateur (`@sum`) sur zéro, exécutez une instruction `INSERT`, puis vérifiez la valeur de la variable:

```
mysql> SET @sum = 0;  
mysql> INSERT INTO account VALUES(137,14.98), (141,1937.50), (97,-100.00);  
mysql> SELECT @sum AS 'Total amount inserted';  
+-----+  
| Total amount inserted |  
+-----+  
| 1852.48                |  
+-----+
```

Dans ce cas, la valeur de `@sum` après l'exécution de l'instruction `INSERT` est  $14.98 + 1937.50 - 100$  ou `1852.48`.

### Déclencheur de chute

```
mysql> DROP TRIGGER test.ins_sum;
```

Si vous déposez une table, tous les déclencheurs de la table sont également supprimés.

## Types de déclencheurs

### Timing

Il existe deux modificateurs de temps d'action du déclencheur:

- **BEFORE** déclencheur ne s'active avant d'exécuter la requête,
- **AFTER** déclenchement du tir après le changement.

---

## Événement déclencheur

Trois déclencheurs peuvent être associés à trois événements:

- INSERT
- UPDATE
- DELETE

---

## Avant Insérer exemple de déclenchement

```
DELIMITER $$

CREATE TRIGGER insert_date
  BEFORE INSERT ON stack
  FOR EACH ROW
BEGIN
  -- set the insert_date field in the request before the insert
  SET NEW.insert_date = NOW();
END;

$$
DELIMITER ;
```

---

## Avant mise à jour exemple de déclenchement

```
DELIMITER $$

CREATE TRIGGER update_date
  BEFORE UPDATE ON stack
  FOR EACH ROW
BEGIN
  -- set the update_date field in the request before the update
  SET NEW.update_date = NOW();
END;

$$
DELIMITER ;
```

---

## Après Supprimer exemple de déclenchement

```
DELIMITER $$

CREATE TRIGGER deletion_date
  AFTER DELETE ON stack
  FOR EACH ROW
```

```
BEGIN
    -- add a log entry after a successful delete
    INSERT INTO log_action(stack_id, deleted_date) VALUES(OLD.id, NOW());
END;

$$
DELIMITER ;
```

Lire TRIGGERS en ligne: <https://riptutorial.com/fr/mysql/topic/3069/triggers>



# Chapitre 69: Types de données

## Exemples

### Moulage implicite / automatique

```
select '123' * 2;
```

Pour faire la **multiplication** avec `2` MySQL convertit automatiquement la chaîne `123` en un nombre.

Valeur de retour:

246

La conversion en nombre commence de gauche à droite. Si la conversion n'est pas possible, le résultat est `0`

```
select '123ABC' * 2
```

Valeur de retour:

246

```
select 'ABC123' * 2
```

Valeur de retour:

0

### VARCHAR (255) - ou non

#### Max suggéré

Tout d'abord, je mentionnerai quelques chaînes communes qui sont toujours hexadécimales ou limitées à ASCII. Pour ceux-ci, vous devez spécifier `CHARACTER SET ascii` (`latin1` est ok) pour qu'il ne gaspille pas d'espace:

```
UUID CHAR(36) CHARACTER SET ascii -- or pack into BINARY(16)
country_code CHAR(2) CHARACTER SET ascii
ip_address CHAR(39) CHARACTER SET ascii -- or pack into BINARY(16)
phone VARCHAR(20) CHARACTER SET ascii -- probably enough to handle extension
postal_code VARCHAR(20) CHARACTER SET ascii -- (not 'zip_code') (don't know the max

city VARCHAR(100) -- This Russian town needs 91:
    Poselok Uchebnogo Khozyaystva Srednego Professionalno-Tekhnicheskoye Uchilishche Nomer
    Odin
country VARCHAR(50) -- probably enough
```

```
name VARCHAR(64) -- probably adequate; more than some government agencies allow
```

**Pourquoi pas simplement 255?** Il y a deux raisons d'éviter la pratique courante d'utiliser (255) pour tout.

- Lorsqu'un `SELECT` complexe doit créer une table temporaire (pour une sous-requête, `UNION`, `GROUP BY`, etc.), le choix préféré consiste à utiliser le moteur `MEMORY`, qui place les données dans la RAM. Mais les `VARCHARs` sont transformés en `CHAR` dans le processus. Cela fait que `VARCHAR(255) CHARACTER SET utf8mb4` prend 1020 octets. Cela peut conduire à devoir renverser sur le disque, ce qui est plus lent.
- Dans certaines situations, InnoDB examinera la taille potentielle des colonnes d'une table et décidera qu'il sera trop gros, abandonnant une `CREATE TABLE`.

## VARCHAR contre TEXTE

Conseils d'utilisation pour `*TEXT`, `CHAR` et `VARCHAR`, ainsi que quelques bonnes pratiques:

- N'utilisez jamais `TINYTEXT`.
- N'utilisez presque jamais `CHAR` - c'est une longueur fixe; chaque caractère correspond à la longueur maximale du `CHARACTER SET` (par exemple, 4 octets / caractère pour `utf8mb4`).
- Avec `CHAR`, utilisez `CHARACTER SET ascii` sauf indication contraire.
- `VARCHAR(n)` tronquera *n caractères*; `TEXT` sera tronqué à un certain nombre d'octets. (Mais voulez-vous une troncature?)
- `*TEXT` peut ralentir les `SELECTs` complexes en raison de la gestion des tables temporaires.

## INT comme AUTO\_INCREMENT

Toute taille de `INT` peut être utilisée pour `AUTO_INCREMENT`. `UNSIGNED` est toujours approprié.

Gardez à l'esprit que certaines opérations "gravent" les identifiants `AUTO_INCREMENT`. Cela pourrait entraîner un écart inattendu. Exemples: `INSERT IGNORE` et `REPLACE`. Ils *peuvent* pré - allouer un identifiant *avant de* réaliser que cela ne sera pas nécessaire. Ceci est un comportement attendu et par conception dans le moteur InnoDB et ne devrait pas décourager leur utilisation.

## Autres

Il existe déjà une entrée distincte pour "FLOAT, DOUBLE et DECIMAL" et "ENUM". Une seule page sur les types de données est susceptible d'être lourde - Je suggère "Types de champs" (ou devrait-il s'appeler "Types de données"?) Soit une vue d'ensemble, puis divisée en deux pages:

- Les INT
- FLOAT, DOUBLE et DECIMAL
- Strings (CHARs, TEXT, etc.)
- BINARY et BLOB
- DATETIME, TIMESTAMP et amis
- ENUM et SET
- Données spatiales
- [Type JSON](#) (MySQL 5.7.8+)

- Comment représenter Money et d'autres «types» courants nécessitant un changement de style dans des types de données existants

Le cas échéant, chaque page de sujet devrait inclure, en plus de la syntaxe et des exemples:

- Considérations lors de la modification
- Taille (octets)
- Contraste avec des moteurs non MySQL (faible priorité)
- Considérations à prendre en compte lors de l'utilisation du type de données dans une clé principale ou secondaire
- autre meilleure pratique
- autres problèmes de performance

(Je suppose que cet "exemple" va se distraire lorsque mes suggestions ont été satisfaites ou ont fait l'objet d'un veto.)

## Introduction (numérique)

MySQL propose un certain nombre de types numériques différents. Ceux-ci peuvent être décomposés en

| Groupe                     | Les types   |
|----------------------------|---|
| Types entiers              | INTEGER , INT , SMALLINT , TINYINT , MEDIUMINT , BIGINT |
| Types de points fixes      | DECIMAL , NUMERIC                                       |
| Types de virgule flottante | FLOAT DOUBLE  |
| Type de valeur de bit      | BIT   |

## Types entiers

La valeur non signée minimale est toujours 0.

| Type      | Espace de rangement (Octets) | Valeur minimum (Signé)  | Valeur maximum (Signé)    | Valeur maximum (Non signé) |
|-----------|------------------------------|-------------------------|---------------------------|----------------------------|
| TINYINT   | 1                            | $-2^7$<br>-128          | $2^7 - 1$<br>127          | $2^8 - 1$<br>255           |
| SMALLINT  | 2                            | $-2^{15}$<br>-32 768    | $2^{15} - 1$<br>32 767    | $2^{16} - 1$<br>65 535     |
| MEDIUMINT | 3                            | $-2^{23}$<br>-8,388,608 | $2^{23} - 1$<br>8,388,607 | $2^{24} - 1$<br>16,777,215 |

| Type   | Espace de rangement (Octets) | Valeur minimum (Signé)                      | Valeur maximum (Signé)                       | Valeur maximum (Non signé)                    |
|--------|------------------------------|---|--|---|
| INT    | 4                            | $-2^{31}$<br>-2 147 483 648                 | $2^{31} - 1$<br>2 147 483 647                | $2^{32} - 1$<br>4 294 967 295                 |
| BIGINT | 8                            | $-2^{63}$<br>-<br>9.223.372.036.854.775.808 | $2^{63} - 1$<br>9 223 372 036<br>854 775 807 | $2^{64} - 1$<br>18 446 744 073<br>709 551 615 |

## Types de points fixes

Les types MySQL `DECIMAL` et `NUMERIC` stockent des valeurs de données numériques exactes. Il est recommandé d'utiliser ces types pour conserver une précision exacte, comme pour l'argent.

### Décimal

Ces valeurs sont stockées au format binaire. Dans une déclaration de colonne, la précision et l'échelle doivent être spécifiées

La précision représente le nombre de chiffres significatifs stockés pour les valeurs.

L'échelle représente le nombre de chiffres stockés *après* la décimale

```
salary DECIMAL(5,2)
```

5 représente la *precision* et 2 représente l' *scale* . Pour cet exemple, la plage de valeurs pouvant être stockée dans cette colonne est `-999.99 to 999.99` entre `-999.99 to 999.99`

Si le paramètre d'échelle est omis, sa valeur par défaut est 0

Ce type de données peut stocker jusqu'à 65 chiffres.

Le nombre d'octets pris par `DECIMAL(M,N)` est *approximativement de*  $M/2$  .

## Types de virgule flottante

`FLOAT` et `DOUBLE` représentent des types de données *approximatifs* .

| Type   | Espace de rangement | Précision                                      | Gamme         |
|--------|---------------------|--|---------------|
| FLOTTE | 4 octets            | 23 bits significatifs / ~ 7 chiffres décimaux  | $10^{+/-38}$  |
| DOUBLE | 8 octets            | 53 bits significatifs / ~ 16 chiffres décimaux | $10^{+/-308}$ |

`REAL` est un synonyme de `FLOAT`. `DOUBLE PRECISION` est un synonyme de `DOUBLE`.

Bien que qualifié MySQL permet également (M, D), ne l'utilisez pas. (M, D) signifie que les valeurs peuvent être stockées avec un maximum de M chiffres, où D peut être après la décimale. *Les nombres seront arrondis deux fois ou tronqués; Cela causera plus de problèmes que d'avantages.*

Étant donné que les valeurs à virgule flottante sont approximatives et ne sont pas stockées en tant que valeurs exactes, les tentatives de les traiter comme exactes dans les comparaisons peuvent entraîner des problèmes. Notez en particulier qu'une valeur `FLOAT` est rarement égale à une valeur `DOUBLE`.

## Type de valeur de bit

Le type `BIT` est utile pour stocker les valeurs de champ binaire. `BIT(M)` permet de stocker jusqu'à M bits, M étant compris entre 1 to 64

Vous pouvez également spécifier des valeurs avec `bit value notation de bit value`.

```
b'111'      -> 7
b'10000000' -> 128
```

Parfois, il est utile d'utiliser «shift» pour construire une valeur à un seul bit, par exemple `(1 << 7)` pour 128.

La taille maximale combinée de toutes les colonnes BIT d'une table `NDB` est 4096.

## CHAR (n)

`CHAR(n)` est une chaîne d'une longueur fixe de n caractères. S'il s'agit de `CHARACTER SET utf8mb4`, cela signifie qu'il occupe exactement `4*n octets`, quel que soit le texte qu'il `CHARACTER SET utf8mb4`.

La plupart des cas d'utilisation de `CHAR(n)` impliquent des chaînes contenant des caractères anglais, par conséquent devraient être `CHARACTER SET ascii`. (`latin1` fera tout aussi bien.)

```
country_code CHAR(2) CHARACTER SET ascii,
postal_code  CHAR(6) CHARACTER SET ascii,
uuid         CHAR(39) CHARACTER SET ascii, -- more discussion elsewhere
```

## DATE, DATETIME, TIMESTAMP, YEAR et TIME

Le type de données `DATE` comprend la date mais pas de composant temporel. Son format est `'YYYY-MM-DD'` avec une plage de «1000-01-01» à «9999-12-31».

Le type `DATETIME` inclut l'heure au format «AAAA-MM-JJ HH: MM: SS». Il est compris entre '1000-01-01 00:00:00' et '9999-12-31 23:59:59'.

Le type `TIMESTAMP` est un type entier comprenant la date et l'heure avec une plage effective de '1970-01-01 00:00:01' UTC à '2038-01-19 03:14:07' UTC.

Le type `YEAR` représente une année et se situe entre 1901 et 2155.

Le type `TIME` représente une heure au format 'HH: MM: SS' et contient une plage allant de '-838: 59: 59' à '838: 59: 59'.

---

Exigences de stockage:

| Data Type | Before MySQL 5.6.4 | as of MySQL 5.6.4                    |
|-----------|--------------------|--------------------------------------|
| YEAR      | 1 byte             | 1 byte                               |
| DATE      | 3 bytes            | 3 bytes                              |
| TIME      | 3 bytes            | 3 bytes + fractional seconds storage |
| DATETIME  | 8 bytes            | 5 bytes + fractional seconds storage |
| TIMESTAMP | 4 bytes            | 4 bytes + fractional seconds storage |

Fractional Seconds (à partir de la version 5.6.4):

| Fractional Seconds Precision | Storage Required |
|------------------------------|------------------|
| 0                            | 0 bytes          |
| 1,2                          | 1 byte           |
| 3,4                          | 2 byte           |
| 5,6                          | 3 byte           |

Voir les [types de pages DATE, DATETIME et TIMESTAMP](#) , les exigences de stockage des types de données et les fractions de secondes dans les valeurs temporelles du manuel MySQL.

Lire Types de données en ligne: <https://riptutorial.com/fr/mysql/topic/4137/types-de-donnees>

# Chapitre 70: Un à plusieurs

## Introduction

L'idée de un à plusieurs (1: M) concerne la réunion des lignes les unes avec les autres, en particulier les cas où une seule ligne d'une table correspond à plusieurs lignes d'une autre.

1: M est unidirectionnel, c'est-à-dire que chaque fois que vous interrogez une relation 1: M, vous pouvez utiliser la ligne "one" pour sélectionner "plusieurs" lignes dans une autre table, mais vous ne pouvez pas utiliser une seule ligne "many". sélectionnez plus d'une seule ligne.

## Remarques

Pour la plupart des cas, travailler avec une relation 1: M nécessite de comprendre *les clés primaires* et *les clés étrangères* .

**Une clé primaire** est une colonne dans une table où une seule ligne de cette colonne représente une seule entité ou, la sélection d'une valeur dans une colonne de clé primaire entraîne une seule ligne. En utilisant les exemples ci-dessus, un EMP\_ID représente un seul employé. Si vous interrogez un seul EMP\_ID, vous verrez une seule ligne représentant l'employé correspondant.

**Une clé étrangère** est une colonne d'une table qui correspond à la clé primaire d'une autre table différente. Dans notre exemple ci-dessus, le MGR\_ID de la table EMPLOYEES est une clé étrangère. Généralement, pour joindre deux tables, vous les joindrez en fonction de la clé primaire d'une table et de la clé étrangère dans une autre.

## Exemples

### Exemples de tables d'entreprise

Considérez une entreprise où chaque employé qui est un gestionnaire gère un ou plusieurs employés et chaque employé a un seul gestionnaire.

Cela se traduit par deux tables:

#### DES EMPLOYÉS

| EMP_ID | PRÉNOM | NOM DE FAMILLE             | MGR_ID |
|--------|--------|----------------------------|--------|
| E01    | Johnny | Appleseed                  | M02    |
| E02    | Erin   | Macklemore                 | M01    |
| E03    | Colby  | Formalités administratives | M03    |
| E04    | Ron    | Sonswan                    | M01    |

## GESTIONNAIRES

| MGR_ID | PRÉNOM      | NOM DE FAMILLE |
|--------|-------------|----------------|
| M01    | Bruyant     | McQueen        |
| M02    | Autoritaire | Un pantalon    |
| M03    | Baril       | Jones          |

## Gérer les employés par un seul responsable

```
SELECT e.emp_id , e.first_name , e.last_name FROM employees e INNER JOIN managers m ON m.mgr_id = e.mgr_id WHERE m.mgr_id = 'M01' ;
```

Résulte en:

| EMP_ID | PRÉNOM | NOM DE FAMILLE |
|--------|--------|----------------|
| E02    | Erin   | Macklemore     |
| E04    | Ron    | Sonswan        |

En fin de compte, pour chaque gestionnaire que nous recherchons, nous verrons un ou plusieurs employés retournés.

## Obtenir le gestionnaire pour un seul employé

Consultez les exemples de tableaux ci-dessus en regardant cet exemple.

```
SELECT m.mgr_id , m.first_name , m.last_name FROM managers m INNER JOIN employees e ON e.mgr_id = m.mgr_id WHERE e.emp_id = 'E03' ;
```

| MGR_ID | PRÉNOM | NOM DE FAMILLE |
|--------|--------|----------------|
| M03    | Baril  | Jones          |

Comme c'est l'inverse de l'exemple ci-dessus, nous savons que pour chaque employé recherché, nous ne verrons qu'un seul responsable.

Lire Un à plusieurs en ligne: <https://riptutorial.com/fr/mysql/topic/9600/un-a-plusieurs>



# Chapitre 71: Unions MySQL

## Syntaxe

- `SELECT nom_colonne (s) FROM table1 UNION SELECT nom_colonne (s) FROM table2;`
- `SELECT nom_colonne (s) FROM table1 UNION ALL SELECT nom_colonne (s) FROM table2;`
- `SELECT nom_colonne (s) FROM table1 WHERE nom_colonne = "XYZ" UNION ALL SELECT nom_colonne (s) FROM table2 WHERE nom_colonne = "XYZ";`

## Remarques

`UNION DISTINCT` est identique à `UNION` ; il est plus lent que `UNION ALL` cause d'une passe de déduplication. Une bonne pratique est de toujours épeler `DISTINCT` ou `ALL` , signalant ainsi que vous avez pensé à quoi faire.

## Exemples

### Opérateur syndical

L'opérateur `UNION` est utilisé pour combiner le résultat ( *uniquement des valeurs distinctes* ) de plusieurs instructions `SELECT`.

**Requête:** (Pour sélectionner toutes les villes différentes ( *uniquement des valeurs distinctes* ) dans les tables "Clients" et "Fournisseurs")

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

### Résultat:

```
Number of Records: 10
```

```
City
-----
Aachen
Albuquerque
Anchorage
Annecy
Barcelona
Barquisimeto
Bend
Bergamo
Berlin
Bern
```

## Union TOUT

UNION ALL pour sélectionner toutes les villes (valeurs dupliquées également) dans les tables "Customers" et "Suppliers".

Question:

```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
ORDER BY City;
```

Résultat:

Number of Records: 12

```
City
-----
Aachen
Albuquerque
Anchorage
Ann Arbor
Annecy
Barcelona
Barquisimeto
Bend
Bergamo
Berlin
Berlin
Bern
```

## UNION ALL With WHERE

UNION ALL pour sélectionner toutes les villes allemandes à partir des tables "Clients" et "Fournisseurs". Here `Country="Germany"` doit être spécifié dans la clause where.

Question:

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION ALL
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

Résultat:

Number of Records: 14

| Ville           | Pays      |
|-----------------|-----------|
| Aix-la-Chapelle | Allemagne |

|              |           |
|--------------|-----------|
| Berlin       | Allemagne |
| Berlin       | Allemagne |
| Brandebourg  | Allemagne |
| Cunewalde    | Allemagne |
| Cuxhaven     | Allemagne |
| Francfort    | Allemagne |
| Francfort aM | Allemagne |
| Köln         | Allemagne |
| Leipzig      | Allemagne |
| Mannheim     | Allemagne |
| München      | Allemagne |
| Münster      | Allemagne |
| Stuttgart    | Allemagne |

Lire Unions MySQL en ligne: <https://riptutorial.com/fr/mysql/topic/5376/unions-mysql>

# Chapitre 72: Utiliser des variables

## Exemples

### Définition des variables

Voici quelques façons de définir des variables:

1. Vous pouvez définir une variable sur une chaîne spécifique, un numéro, une date en utilisant SET

EX: SET @var\_string = 'my\_var'; SET @var\_num = '2' SET @var\_date = '2015-07-20';

2. vous pouvez définir une variable pour qu'elle soit le résultat d'une instruction select en utilisant: =

EX: Sélectionnez @var: = '123'; (Remarque: vous devez utiliser: = lors de l'attribution d'une variable n'utilisant pas la syntaxe SET, car dans d'autres instructions, (select, update ...) le "=" est utilisé pour comparer, donc lorsque vous ajoutez un deux-points = ", vous dites" Ce n'est pas une comparaison, c'est un SET ".)

3. Vous pouvez définir une variable pour qu'elle soit le résultat d'une instruction select utilisant INTO

(C'était particulièrement utile lorsque j'avais besoin de choisir dynamiquement les partitions à interroger)

EX: SET @start\_date = '2015-07-20'; SET @end\_date = '2016-01-31';

```
#this gets the year month value to use as the partition names
SET @start_yearmonth = (SELECT EXTRACT(YEAR_MONTH FROM @start_date));
SET @end_yearmonth = (SELECT EXTRACT(YEAR_MONTH FROM @end_date));

#put the partitions into a variable
SELECT GROUP_CONCAT(partition_name)
FROM information_schema.partitions p
WHERE table_name = 'partitioned_table'
AND SUBSTRING_INDEX(partition_name,'P',-1) BETWEEN @start_yearmonth AND @end_yearmonth
INTO @partitions;

#put the query in a variable. You need to do this, because mysql did not recognize my variable
as a variable in that position. You need to concat the value of the variable together with the
rest of the query and then execute it as a stmt.
SET @query =
CONCAT('CREATE TABLE part_of_partitioned_table (PRIMARY KEY(id))
SELECT partitioned_table.*
FROM partitioned_table PARTITION(' , @partitions, ' )
JOIN users u USING(user_id)
WHERE date(partitioned_table.date) BETWEEN ' , @start_date, ' AND ' , @end_date);

#prepare the statement from @query
PREPARE stmt FROM @query;
```

```
#drop table
DROP TABLE IF EXISTS tech.part_of_partitioned_table;
#create table using statement
EXECUTE stmt;
```

## Numéro de ligne et groupe en utilisant des variables dans Select Statement

Disons que nous avons une table `team_person` comme ci-dessous:

```
+=====+=====+
| team |    person |
+=====+=====+
|  A  |    John  |
+-----+-----+
|  B  |    Smith |
+-----+-----+
|  A  |    Walter|
+-----+-----+
|  A  |    Louis |
+-----+-----+
|  C  | Elizabeth|
+-----+-----+
|  B  |    Wayne |
+-----+-----+
```

```
CREATE TABLE team_person AS SELECT 'A' team, 'John' person
UNION ALL SELECT 'B' team, 'Smith' person
UNION ALL SELECT 'A' team, 'Walter' person
UNION ALL SELECT 'A' team, 'Louis' person
UNION ALL SELECT 'C' team, 'Elizabeth' person
UNION ALL SELECT 'B' team, 'Wayne' person;
```

Pour sélectionner la table `team_person` avec une colonne `row_number` supplémentaire, soit

```
SELECT @row_no := @row_no+1 AS row_number, team, person
FROM team_person, (SELECT @row_no := 0) t;
```

OU

```
SET @row_no := 0;
SELECT @row_no := @row_no + 1 AS row_number, team, person
FROM team_person;
```

va sortir le résultat ci-dessous:

```
+=====+=====+=====+
| row_number | team |    person |
+=====+=====+=====+
|          1 |  A  |    John  |
+-----+-----+-----+
|          2 |  B  |    Smith |
+-----+-----+-----+
|          3 |  A  |    Walter |
+-----+-----+-----+
```

```

|          4 | A | Louis |
+-----+-----+-----+
|          5 | C | Elizabeth |
+-----+-----+-----+
|          6 | B | Wayne |
+-----+-----+-----+

```

Enfin, si l' on veut obtenir le `row_number` groupe par colonne `team`

```

SELECT @row_no := IF(@prev_val = t.team, @row_no + 1, 1) AS row_number
      ,@prev_val := t.team AS team
      ,t.person
FROM team_person t,
      (SELECT @row_no := 0) x,
      (SELECT @prev_val := '') y
ORDER BY t.team ASC,t.person DESC;

```

```

+-----+-----+-----+
| row_number | team | person |
+-----+-----+-----+
|          1 | A | Walter |
+-----+-----+-----+
|          2 | A | Louis |
+-----+-----+-----+
|          3 | A | John |
+-----+-----+-----+
|          1 | B | Wayne |
+-----+-----+-----+
|          2 | B | Smith |
+-----+-----+-----+
|          1 | C | Elizabeth |
+-----+-----+-----+

```

Lire Utiliser des variables en ligne: <https://riptutorial.com/fr/mysql/topic/5013/utiliser-des-variables>

# Chapitre 73: VUE

## Syntaxe

- `CREATE VIEW nom_vue AS SELECT nom_colonne (s) FROM nom_table WHERE condition; ///` Syntaxe de création simple
- `CRÉER [OU REMPLACER] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}] [DEFINER = {utilisateur | CURRENT_USER}] [SÉCURITÉ SQL {DEFINER | INVOKER}] VIEW nom_vue [(liste_colonnes)] AS select_statement [AVEC [CASCADED | LOCAL] OPTION DE VÉRIFICATION]; ///` Syntaxe complète de la vue
- `DROP VIEW [SI EXISTE] [nom_base.] Nom_vue; ///` Syntaxe de la vue déroulante

## Paramètres

| Paramètres         | Détails  |
|--------------------|--|
| nom_vue            | Nom de la vue  |
| Instruction SELECT | Instructions SQL à emballer dans les vues. Il peut s'agir d'une instruction SELECT pour extraire des données d'une ou de plusieurs tables. |

## Remarques

Les vues sont des tables virtuelles et ne contiennent pas les données renvoyées. Ils peuvent vous éviter d'écrire des requêtes complexes à plusieurs reprises.

- **Avant qu'une vue soit faite**, sa spécification consiste entièrement en une `SELECT`. L'`SELECT` ne peut pas contenir de sous-requête dans la clause `FROM`.
- **Une fois vue est fait**, il est utilisé en grande partie comme une table et peut être `SELECT` ed d' un peu comme une table.

Vous devez créer des vues, lorsque vous souhaitez restreindre quelques colonnes de votre table, à partir de l'autre utilisateur.

- Par exemple: Dans votre organisation, vous voulez que vos responsables consultent peu d'informations à partir d'une table nommée «Ventes», mais vous ne voulez pas que vos ingénieurs logiciels puissent voir tous les champs de la table «Ventes». Ici, vous pouvez créer deux vues différentes pour vos responsables et vos ingénieurs logiciels.

**Performance** `VIEWS` sont du sucre syntaxique. Cependant, les performances peuvent être inférieures ou égales à la requête équivalente avec la sélection de la vue intégrée. L'Optimiseur tente de faire cela "plié" pour vous, mais ne réussit pas toujours. MySQL 5.7.6 fournit d'autres améliorations dans l'Optimizer. Mais, peu importe, en utilisant une `VIEW` ne génère pas une

requête *plus rapide*.

## Exemples

### Créer une vue

#### Les privilèges

L'instruction `CREATE VIEW` requiert le privilège `CREATE VIEW` pour la vue et certains privilèges pour chaque colonne sélectionnée par l'instruction `SELECT`. Pour les colonnes utilisées ailleurs dans l'instruction `SELECT`, vous devez disposer du privilège `SELECT`. Si la clause `OR REPLACE` est présente, vous devez également disposer du privilège `DROP` pour la vue. `CREATE VIEW` peut également nécessiter le privilège `SUPER`, en fonction de la valeur `DEFINER`, comme décrit plus loin dans cette section.

Lorsqu'une vue est référencée, la vérification des privilèges se produit.

Une vue appartient à une base de données. Par défaut, une nouvelle vue est créée dans la base de données par défaut. Pour créer la vue explicitement dans une base de données donnée, utilisez un nom complet

Par exemple:

`db_name.view_name`

```
mysql> CREATE VIEW test.v AS SELECT * FROM t;
```

*Remarque - Dans une base de données, les tables de base et les vues partagent le même espace de noms. Par conséquent, une table de base et une vue ne peuvent pas porter le même nom.*

Une vue peut:

- être créé à partir de nombreux types d'instructions `SELECT`
- se référer aux tables de base ou à d'autres vues
- utiliser des jointures, `UNION` et des sous-requêtes
- `SELECT` n'a même pas besoin de faire référence à des tables

#### Un autre exemple

L'exemple suivant définit une vue qui sélectionne deux colonnes d'une autre table, ainsi qu'une expression calculée à partir de ces colonnes:

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;
mysql> SELECT * FROM v;
+-----+-----+-----+
| qty  | price | value |
+-----+-----+-----+
| 3    | 50    | 150   |
+-----+-----+-----+
```



## Restrictions

- Avant MySQL 5.7.7, l'instruction SELECT ne pouvait pas contenir de sous-requête dans la clause FROM.
- L'instruction SELECT ne peut pas faire référence à des variables système ou à des variables définies par l'utilisateur.
- Dans un programme stocké, l'instruction SELECT ne peut pas faire référence à des paramètres de programme ou à des variables locales.
- L'instruction SELECT ne peut pas faire référence à des paramètres d'instruction préparés.
- Toute table ou vue mentionnée dans la définition doit exister. Une fois la vue créée, il est possible de supprimer une table ou une vue la définition se réfère à. Dans ce cas, l'utilisation de la vue entraîne une erreur. Pour vérifier une définition de vue pour des problèmes de ce type, utilisez l'instruction CHECK TABLE.
- La définition ne peut pas faire référence à une table TEMPORARY et vous ne pouvez pas créer une vue TEMPORAIRE.
- Vous ne pouvez pas associer un déclencheur à une vue.
- Les alias pour les noms de colonne dans l'instruction SELECT sont vérifiés par rapport à la longueur de colonne maximale de 64 caractères (pas l'alias maximum longueur de 256 caractères).
- Une VIEW peut ou ne peut pas optimiser ainsi que l'équivalent SELECT . Il est peu probable d'optimiser mieux.

## Une vue de deux tables

Une vue est très utile lorsqu'elle peut être utilisée pour extraire des données de plusieurs tables.

```
CREATE VIEW myview AS
SELECT a.*, b.extra_data FROM main_table a
LEFT OUTER JOIN other_table b
ON a.id = b.id
```

Dans mysql, les vues ne sont pas matérialisées. Si vous exécutez maintenant la requête simple `SELECT * FROM myview`, mysql exécutera réellement la LEFT JOIN en arrière-plan.

Une vue une fois créée peut être jointe à d'autres vues ou tableaux

## Mise à jour d'une table via une vue

A VIEW comporte comme une table. Bien que vous pouvez UPDATE À UPDATE une table, vous pouvez ou ne pas être en mesure de mettre à jour une vue sur cette table. En général, si le SELECT dans la vue est suffisamment complexe pour nécessiter une table temporaire, alors UPDATE n'est pas autorisé.

Des éléments tels que GROUP BY , UNION , HAVING , DISTINCT et certaines sous-requêtes empêchent la mise à jour de la vue. Détails dans le [manuel de référence](#) .

## DROPPING A VIEW

- Créer et supprimer une vue dans la base de données actuelle.

```
CREATE VIEW few_rows_from_t1 AS SELECT * FROM t1 LIMIT 10;  
DROP VIEW few_rows_from_t1;
```

- Créer et supprimer une vue référençant une table dans une base de données différente.

```
CREATE VIEW table_from_other_db AS SELECT x FROM db1.foo WHERE x IS NOT NULL;  
DROP VIEW table_from_other_db;
```

Lire VUE en ligne: <https://riptutorial.com/fr/mysql/topic/1489/vue>

# Crédits

| S. No | Chapitres   | Contributeurs  |
|-------|---|--|
| 1     | Démarrer avec MySQL   | <a href="#">A. Raza</a> , <a href="#">Aman Dhanda</a> , <a href="#">Andy</a> , <a href="#">Athafoud</a> , <a href="#">CodeWarrior</a> , <a href="#">Community</a> , <a href="#">Configure</a> , <a href="#">Dipen Shah</a> , <a href="#">e4c5</a> , <a href="#">Epodax</a> , <a href="#">Giacomo Garabello</a> , <a href="#">greatwolf</a> , <a href="#">inetphantom</a> , <a href="#">JayRizzo</a> , <a href="#">juergen d</a> , <a href="#">Lahiru Ashan</a> , <a href="#">Lambda Ninja</a> , <a href="#">Magisch</a> , <a href="#">Marek Skiba</a> , <a href="#">Md. Nahiduzzaman Rose</a> , <a href="#">moopet</a> , <a href="#">msohng</a> , <a href="#">Noah van der Aa</a> , <a href="#">O. Jones</a> , <a href="#">OverCoder</a> , <a href="#">Panda</a> , <a href="#">Parth Patel</a> , <a href="#">rap-2-h</a> , <a href="#">rhavendc</a> , <a href="#">Romain Vincent</a> , <a href="#">YCF_L</a> |
| 2     | ALTER TABLE   | <a href="#">e4c5</a> , <a href="#">JohnLBevan</a> , <a href="#">kolunar</a> , <a href="#">LiuYan</a> , <a href="#">Matas Vaitkevicius</a> , <a href="#">mayojava</a> , <a href="#">Rick James</a> , <a href="#">Steve Chambers</a> , <a href="#">Thuta Aung</a> , <a href="#">WAF</a> , <a href="#">YCF_L</a>  |
| 3     | Arithmétique  | <a href="#">Barranka</a> , <a href="#">Dinidu</a> , <a href="#">Drew</a> , <a href="#">JonMark Perry</a> , <a href="#">O. Jones</a> , <a href="#">RamenChef</a> , <a href="#">Richard Hamilton</a> , <a href="#">Rick James</a>  |
| 4     | Changer le mot de passe   | <a href="#">e4c5</a> , <a href="#">Hardik Kanjariya</a> <sup>٧</sup> , <a href="#">Rick James</a> , <a href="#">Viktor</a> , <a href="#">ydaetskcoR</a>  |
| 5     | CHARGER L'INFILE DE DONNÉES   | <a href="#">aries12</a> , <a href="#">Asaph</a> , <a href="#">bhrached</a> , <a href="#">CGritton</a> , <a href="#">e4c5</a> , <a href="#">RamenChef</a> , <a href="#">Rick James</a> , <a href="#">WAF</a>  |
| 6     | Client MySQL  | <a href="#">Batsu</a> , <a href="#">Nathaniel Ford</a> , <a href="#">Rick James</a>  |
| 7     | Codes d'erreur  | <a href="#">Drew</a> , <a href="#">e4c5</a> , <a href="#">juergen d</a> , <a href="#">Lucas Paolillo</a> , <a href="#">O. Jones</a> , <a href="#">Ponnarasu</a> , <a href="#">Rick James</a> , <a href="#">WAF</a> , <a href="#">Wojciech Kazior</a>   |
| 8     | COMMANDÉ PAR  | <a href="#">Florian Genser</a> , <a href="#">Rick James</a>  |
| 9     | Commentaire Mysql   | <a href="#">Franck Dernoncourt</a> , <a href="#">Rick James</a> , <a href="#">WAF</a> , <a href="#">YCF_L</a>  |
| 10    | Configuration de la connexion SSL                                   | <a href="#">4444</a> , <a href="#">a coder</a> , <a href="#">Eugene</a>  |
| 11    | Configuration et réglage  | <a href="#">ChintaMoney</a> , <a href="#">CodeWarrior</a> , <a href="#">Epodax</a> , <a href="#">Eugene</a> , <a href="#">jan_kiran</a> , <a href="#">Rick James</a>   |
| 12    | Connexion avec UTF-8 en utilisant divers langages de programmation. | <a href="#">Epodax</a> , <a href="#">Rick James</a>  |
| 13    | Conversion de MyISAM à InnoDB                                       | <a href="#">Ponnarasu</a> , <a href="#">Rick James</a> , <a href="#">yukoff</a>  |

|    |  |   |
|----|--|---|
| 14 | Création de bases de données   | <a href="#">Daniel Käfer</a> , <a href="#">Drew</a> , <a href="#">Ponnarasu</a> , <a href="#">R.K123</a> , <a href="#">Rick James</a> , <a href="#">still_learning</a>  |
| 15 | Création de table  | <a href="#">4444</a> , <a href="#">Alex Shesterov</a> , <a href="#">alex9311</a> , <a href="#">andygeers</a> , <a href="#">Aryo</a> , <a href="#">Asaph</a> , <a href="#">Barranka</a> , <a href="#">Benvorth</a> , <a href="#">Brad Larson</a> , <a href="#">CPHPython</a> , <a href="#">Darwin von Corax</a> , <a href="#">Dinidu</a> , <a href="#">Drew</a> , <a href="#">fedorqui</a> , <a href="#">HCarrasko</a> , <a href="#">Jean Vitor</a> , <a href="#">John M</a> , <a href="#">Matt</a> , <a href="#">Misa Lazovic</a> , <a href="#">Panda</a> , <a href="#">Parth Patel</a> , <a href="#">Paulo Freitas</a> , <a href="#">Přemysl Šťastný</a> , <a href="#">Rick</a> , <a href="#">Rick James</a> , <a href="#">Ronnie Wang</a> , <a href="#">Saroj Sasmal</a> , <a href="#">Sebastian Brosch</a> , <a href="#">skytreader</a> , <a href="#">Stefan Rogin</a> , <a href="#">Strawberry</a> , <a href="#">Timothy</a> , <a href="#">ultrajohn</a> , <a href="#">user6655061</a> , <a href="#">vijaykumar</a> , <a href="#">Vini.g.fer</a> , <a href="#">Vladimir Kovpak</a> , <a href="#">WAF</a> , <a href="#">YCF_L</a> , <a href="#">Yury Fedorov</a> |
| 16 | Créer un nouvel utilisateur  | <a href="#">Aminadav</a> , <a href="#">Batsu</a> , <a href="#">Hardik Kanjariya</a> <sup>٧</sup> , <a href="#">josinalvo</a> , <a href="#">Rick James</a> , <a href="#">WAF</a>   |
| 17 | EFFACER  | <a href="#">Batsu</a> , <a href="#">Drew</a> , <a href="#">e4c5</a> , <a href="#">ForguesR</a> , <a href="#">gabe3886</a> , <a href="#">Khurram</a> , <a href="#">Parth Patel</a> , <a href="#">Ponnarasu</a> , <a href="#">Rick James</a> , <a href="#">strangeqargo</a> , <a href="#">WAF</a> , <a href="#">whrrgarbl</a> , <a href="#">уpercube</a> , <a href="#">Илья Плотников</a>   |
| 18 | ENUM   | <a href="#">Philipp</a> , <a href="#">Rick James</a>  |
| 19 | Erreur 1055: ONLY_FULL_GROUP_BY: quelque chose n'est pas dans la clause GROUP BY ... | <a href="#">Damian Yerrick</a> , <a href="#">O. Jones</a>   |
| 20 | Événements   | <a href="#">Drew</a> , <a href="#">rene</a>   |
| 21 | Expressions régulières   | <a href="#">user2314737</a> , <a href="#">YCF_L</a>   |
| 22 | Extraire les valeurs du type JSON  | <a href="#">MohaMad</a>   |
| 23 | Fichiers journaux  | <a href="#">Drew</a> , <a href="#">Rick James</a>   |
| 24 | Groupage   | <a href="#">Drew</a> , <a href="#">Rick James</a>   |
| 25 | Index et clés  | <a href="#">Alex Recarey</a> , <a href="#">Barranka</a> , <a href="#">Ben Visness</a> , <a href="#">Drew</a> , <a href="#">kolunar</a> , <a href="#">Rick James</a> , <a href="#">Sanjeev kumar</a>   |
| 26 | Informations sur le serveur  | <a href="#">FMashiro</a>  |
| 27 | INSÉRER  | <a href="#">0x49D1</a> , <a href="#">AbcAeffchen</a> , <a href="#">Abubakkar</a> , <a href="#">Aukhan</a> , <a href="#">CGritton</a> , <a href="#">Dinidu</a> , <a href="#">Dreamer</a> , <a href="#">Drew</a> , <a href="#">e4c5</a> , <a href="#">fnkr</a> , <a href="#">gabe3886</a> , <a href="#">Horen</a> , <a href="#">Hugo Buff</a> , <a href="#">Ian Kenney</a> , <a href="#">Johan</a> , <a href="#">Magisch</a> , <a href="#">NEER</a> , <a href="#">Parth Patel</a> , <a href="#">Philipp</a> , <a href="#">Rick James</a> , <a href="#">Riho</a> , <a href="#">strangeqargo</a> , <a href="#">Thuta Aung</a> , <a href="#">zeppelin</a>  |

|    |  |   |
|----|--|---|
| 28 | Installez le conteneur Mysql avec Docker-Compose | <a href="#">Marc Alff</a> , <a href="#">molavec</a>   |
| 29 | Jeux de caractères et classements                | <a href="#">frlan</a> , <a href="#">Rick</a> , <a href="#">Rick James</a>   |
| 30 | JOINS: Rejoignez 3 table avec le même nom d'id.  | <a href="#">FMashiro</a>  |
| 31 | Joint  | <a href="#">Artisan72</a> , <a href="#">Batsu</a> , <a href="#">Benvorth</a> , <a href="#">Bikash P</a> , <a href="#">Drew</a> , <a href="#">Matt</a> , <a href="#">Philipp</a> , <a href="#">Rick</a> , <a href="#">Rick James</a> , <a href="#">user3617558</a>   |
| 32 | JSON   | <a href="#">A. Raza</a> , <a href="#">Ben</a> , <a href="#">Drew</a> , <a href="#">e4c5</a> , <a href="#">Manatax</a> , <a href="#">Mark Amery</a> , <a href="#">MohaMad</a> , <a href="#">phatfingers</a> , <a href="#">Rick James</a> , <a href="#">sunkuet02</a>   |
| 33 | L'optimisation des performances                  | <a href="#">e4c5</a> , <a href="#">RamenChef</a> , <a href="#">Rick James</a>   |
| 34 | Limite et décalage                               | <a href="#">Alvaro Flaño Larrondo</a> , <a href="#">Ani Menon</a> , <a href="#">animuson</a> , <a href="#">ChaoticTwist</a> , <a href="#">Chris Rasys</a> , <a href="#">CPHPython</a> , <a href="#">Ian Gregory</a> , <a href="#">Matt S</a> , <a href="#">Rick James</a> , <a href="#">Sumit Gupta</a> , <a href="#">WAF</a> |
| 35 | Manipulation des fuseaux horaires                | <a href="#">O. Jones</a>  |
| 36 | METTRE À JOUR                                    | <a href="#">4thfloorstudios</a> , <a href="#">Chris</a> , <a href="#">Drew</a> , <a href="#">Khurram</a> , <a href="#">Ponnarasu</a> , <a href="#">Rick James</a> , <a href="#">Sevle</a>   |
| 37 | Moteur MyISAM                                    | <a href="#">Rick James</a>  |
| 38 | Mots réservés                                    | <a href="#">juergen d</a> , <a href="#">user2314737</a>   |
| 39 | MySQL Admin                                      | <a href="#">Florian Genser</a> , <a href="#">Matas Vaitkevicius</a> , <a href="#">RationalDev</a> , <a href="#">Rick James</a>  |
| 40 | MySQL LOCK TABLE                                 | <a href="#">Ponnarasu</a> , <a href="#">Rick James</a> , <a href="#">vijeeshin</a>  |
| 41 | Mysql Performance Tips                           | <a href="#">arushi</a> , <a href="#">RamenChef</a> , <a href="#">Rick James</a> , <a href="#">Rodrigo Darti da Costa</a>  |
| 42 | mysqlimport                                      | <a href="#">Batsu</a>   |
| 43 | NUL  | <a href="#">Rick James</a> , <a href="#">Sumit Gupta</a>  |
| 44 | Opérations de chaîne                             | <a href="#">Abubakkar</a> , <a href="#">Batsu</a> , <a href="#">juergen d</a> , <a href="#">kolunar</a> , <a href="#">Rick James</a> , <a href="#">uruloke</a> , <a href="#">WAF</a>  |
| 45 | Opérations de date et heure                      | <a href="#">Abhishek Aggrawal</a> , <a href="#">Drew</a> , <a href="#">Matt S</a> , <a href="#">O. Jones</a> , <a href="#">Rick James</a> , <a href="#">Sumit Gupta</a>   |
| 46 | Par groupe                                       | <a href="#">Adam</a> , <a href="#">Filipe Martins</a> , <a href="#">Lijo</a> , <a href="#">Rick James</a> , <a href="#">Thuta Aung</a> , <a href="#">WAF</a> ,  |

|    |  |   |
|----|--|---|
|    |  | <a href="#">whrrgarbl</a>   |
| 47 | Partitionnement  | <a href="#">Majid</a> , <a href="#">Rick James</a>  |
| 48 | Personnaliser PS1  | <a href="#">Eugene</a> , <a href="#">Wenzhong</a>   |
| 49 | Pointes  | <a href="#">Drew</a> , <a href="#">SuperDJ</a>  |
| 50 | PREPARE Déclarations   | <a href="#">kolunar</a> , <a href="#">Rick James</a> , <a href="#">winter</a>   |
| 51 | Recherche en texte intégral  | <a href="#">O. Jones</a>  |
| 52 | Récupérer du mot de passe root perdu                                       | <a href="#">BacLuc</a> , <a href="#">Jen R</a>  |
| 53 | Récupérer et réinitialiser le mot de passe root par défaut pour MySQL 5.7+ | <a href="#">Lahiru</a> , <a href="#">ParthaSen</a>  |
| 54 | Réplication  | <a href="#">Ponnarasu</a>   |
| 55 | Requêtes pivot   | <a href="#">Barranka</a>  |
| 56 | Routines stockées (procédures et fonctions)                                | <a href="#">Abhishek Aggrawal</a> , <a href="#">Abubakkar</a> , <a href="#">Darwin von Corax</a> , <a href="#">Dinidu</a> , <a href="#">Drew</a> , <a href="#">e4c5</a> , <a href="#">juergen d</a> , <a href="#">kolunar</a> , <a href="#">llanato</a> , <a href="#">Rick James</a> , <a href="#">userlond</a>   |
| 57 | Sauvegarde avec mysqldump  | <a href="#">agold</a> , <a href="#">Asaph</a> , <a href="#">Barranka</a> , <a href="#">Batsu</a> , <a href="#">KalenGi</a> , <a href="#">Mark Amery</a> , <a href="#">Matthew</a> , <a href="#">mnoronha</a> , <a href="#">Ponnarasu</a> , <a href="#">RamenChef</a> , <a href="#">Rick James</a> , <a href="#">still_learning</a> , <a href="#">strangeqargo</a> , <a href="#">Sumit Gupta</a> , <a href="#">Timothy</a> , <a href="#">WAF</a>   |
| 58 | Sécurité via GRANTS  | <a href="#">Rick James</a>  |
| 59 | SÉLECTIONNER   | <a href="#">Ani Menon</a> , <a href="#">Asjad Athick</a> , <a href="#">Benvorth</a> , <a href="#">Bhavin Solanki</a> , <a href="#">Chip</a> , <a href="#">Drew</a> , <a href="#">greatwolf</a> , <a href="#">Inzimam Tariq IT</a> , <a href="#">julienc</a> , <a href="#">KartikKannapur</a> , <a href="#">Kruti Patel</a> , <a href="#">Matthis Kohli</a> , <a href="#">O. Jones</a> , <a href="#">Ponnarasu</a> , <a href="#">Rick James</a> , <a href="#">SeeuD1</a> , <a href="#">ThisIsImpossible</a> , <a href="#">timmyRS</a> , <a href="#">YCF_L</a> , <a href="#">ypercube</a> |
| 60 | SYNDICAT   | <a href="#">Mattew Whitt</a> , <a href="#">Rick James</a> , <a href="#">Riho</a> , <a href="#">Tarik</a> , <a href="#">wangengzheng</a>   |
| 61 | Table de chute   | <a href="#">Noah van der Aa</a> , <a href="#">Parth Patel</a> , <a href="#">Ponnarasu</a> , <a href="#">R.K123</a> , <a href="#">Rick James</a> , <a href="#">trf</a> , <a href="#">Tushar patel</a> , <a href="#">YCF_L</a>  |
| 62 | Table de mappage plusieurs-à-plusieurs                                     | <a href="#">Rick James</a>  |
| 63 | Tableau dynamique de pivotement à l'aide de                                | <a href="#">rpd</a>   |

|    |   |  |
|----|---|--|
|    | l'instruction préparée                  |  |
| 64 | Tables temporaires                      | <a href="#">Ponnarasu</a> , <a href="#">Rick James</a>   |
| 65 | Temps avec précision de la seconde      | <a href="#">O. Jones</a>   |
| 66 | Traiter des données rares ou manquantes | <a href="#">Batsu</a> , <a href="#">Nate Vaughan</a>   |
| 67 | Transaction                             | <a href="#">Ponnarasu</a> , <a href="#">Rick James</a>   |
| 68 | TRIGGERS                                | <a href="#">Blag</a> , <a href="#">e4c5</a> , <a href="#">Matas Vaitkevicius</a> , <a href="#">ratchet</a> , <a href="#">WAF</a> , <a href="#">YCF_L</a>   |
| 69 | Types de données                        | <a href="#">Batsu</a> , <a href="#">dakab</a> , <a href="#">Drew</a> , <a href="#">Dylan Vander Berg</a> , <a href="#">e4c5</a> , <a href="#">juergen d</a> , <a href="#">MohaMad</a> , <a href="#">Richard Hamilton</a> , <a href="#">Rick James</a>  |
| 70 | Un à plusieurs                          | <a href="#">falsefive</a>  |
| 71 | Unions MySQL                            | <a href="#">Ani Menon</a> , <a href="#">Rick James</a>   |
| 72 | Utiliser des variables                  | <a href="#">kolunar</a> , <a href="#">user6655061</a>  |
| 73 | VUE                                     | <a href="#">Abhishek Aggrawal</a> , <a href="#">Divya</a> , <a href="#">e4c5</a> , <a href="#">Marina K.</a> , <a href="#">Nikita Kurtin</a> , <a href="#">Ponnarasu</a> , <a href="#">R.K123</a> , <a href="#">ratchet</a> , <a href="#">Rick James</a> , <a href="#">WAF</a> , <a href="#">Yury Fedorov</a> , <a href="#">Илья Плотников</a> |