Version: 2020/08/07 16:26

LDF305 - Gestion des Données avec SQL

Contenu du Module

- **LDF305 -La Langage SQL**
  - SQL
    - Chaînes de caractères
    - Nombres
      - Nombres Entiers
      - Nombres Décimaux
      - Nombres Négatifs
    - Valeurs NULL
    - Noms de Fichiers
    - Variables Utilisateurs
    - Commentaires
    - Commandes
      - SELECT
      - UPDATE
      - DELETE FROM
      - DROP TABLE
      - INSERT
      - ALTER
      - MATCH
    - Opérateurs
      - Mathématiques
      - Logiques
      - Comparaison
    - Fonctions
      - Mathématiques

                - Chaînes
                - Dates
                - Contrôle
                - Agrégation
                - Autres
    - Types de Champs
        - Nombres entiers
        - Nombres à virgule flottante
        - Dates et Heures
            - Types de données TIMESTAMP
        - Chaînes
        - TEXT et BLOB
        - ENUM et SET
    - Types de Moteurs de Stockage
    - Caractéristiques des Moteurs
        - InnoDB
            - Mécanisme Interne
            - Transactions
            - Tablespace
            - Multiversion Concurrency Control
            - Transaction Isolation Levels
        - MyISAM
            - MyISAM FIXED
            - MyISAM DYNAMIC
            - MyISAM COMPRESSED
            - Particularités
        - Memory
            - Particularités
            - En Pratique
        - Archive
            - Particularités
        - CSV
        - FEDERATED
        - NDB Cluster

        - Autres Moteurs Non Standards
            - XtraDB
            - Aria
    - Jointures
        - FULL JOIN
        - LEFT JOIN
        - RIGHT JOIN
    - LAB #1 - Le Langage SQL

SQL

Chaînes de caractères

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

Exemples

'Linux est incroyable'

“Windows est …….”

“Je lui dit : ”“Je t'aime”””

Nombres

Il existe 3 types de nombres :

Nombres Entiers

Une séquence de chiffres **sans espaces**

Exemple

999256

0

Nombres Décimaux

Utilisent le **point** comme séparateur.

Exemple

120.54

5566.8956e+12

Nombres Négatifs

Sont précédés par le signe **-**.

Exemple

-458

-147.36

Valeurs NULL

Une chaîne **sans données**.

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

Noms de Fichiers

Les noms de bases, tables et colonnes :

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

Les alias :

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

Variables Utilisateurs

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

Commentaires

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

Les commentaires sous plusieurs lignes sont entourés comme suit :

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

Commandes

SELECT

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

Syntaxe

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

Exemples

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

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

UPDATE

Mettre à jour des données dans une table.

Syntaxe

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

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

Exemples

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

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

DELETE FROM

Supprimer des enregistrements d'une table.

Syntaxe

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

Exemple

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

DROP TABLE

Syntaxe

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

Supprimer une table d'une base de données.

Exemple

```
DROP TABLE Test;
```

INSERT

Syntaxe

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

ou

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

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

INSERT VALUES

Insérer des valeurs dans une table.

Exemple

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

INSERT SELECT

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

Exemple

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

ALTER

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

ALTER ADD

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

Exemples

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

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

ALTER CHANGE

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

Exemple

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

ALTER DROP

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

Exemple

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

ALTER KEYS

Permet de désactiver la clef primaire momentanément.

Exemples

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

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

ALTER RENAME

Permet de renommer une table.

Exemple

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

ALTER ORDER BY

Permet de reclasser physiquement une table.

Exemple

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

MATCH

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

Exemple

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

Opérateurs

Mathémathiques

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

Logiques

Nom	Description
!	NON
||	OU
&&	ET
XOR	XOR exclusif

Comparaison

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

Fonctions

Mathémathiques

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

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

Chaînes

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

Dates

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

Motifs

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

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

Par exemple :

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

mysql>
```

Contrôle

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

Agrégation

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

Autres

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

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

Types de Champs

Nombres entiers

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

Nombres à virgule flottante

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

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

Dates et Heures

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

Types de données TIMESTAMP

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

Chaînes

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

TEXT et BLOB

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

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

ENUM et SET

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

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

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

Types de Moteurs de Stockage

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

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

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

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

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

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

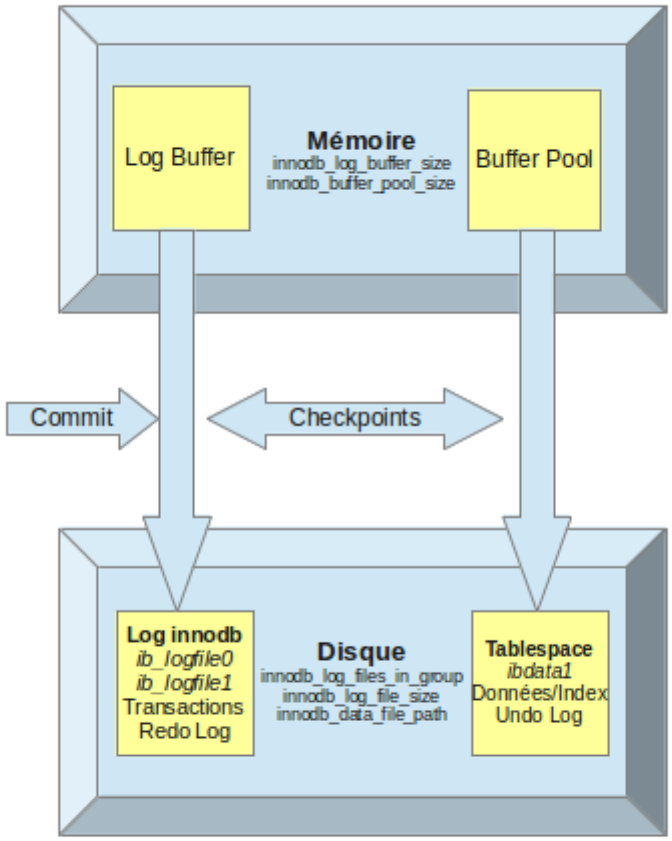
Caractéristiques des Moteurs

InnoDB

Le moteur InnoDB est le moteur par défaut de MySQL depuis la version 5.5. Les principaux caractéristiques du moteur InnoDB sont :

- Il est transactionnel de type **ACID** (Atomicité, Cohérence, Isolation, Durabilité),
- Il implémente le **MVCC** (Multi Version Concurrency Control) qui permet d'avoir des lectures qui ne bloquent pas des écritures et inversement,
- Il implémente un verrou au niveau enregistrement,
- Il implémente un mécanisme de restauration automatique en utilisant des journaux de transactions **ib_logfile*** ou * est **0**, **1**, **2** etc,
- La structure de la table est stockée dans un fichier .frm,
- Il supporte l'utilisation de clefs étrangères,
- Il peut être sauvegardé à chaud,
- Il est recommandé pour des applications avec beaucoup d'écritures.

Mécanisme Interne



Les transactions en cours sont stockées en mémoire dans le **buffer des journaux de transactions** (*log_buffer*). La taille du buffer est fixée par la valeur de l'option **innodb_log_buffer_size**. Chaque seconde les informations dans le buffer sont écrites sur disque dans l'un des deux journaux des transactions **ib_logfile0** ou **ib_logfile1**. Le nombre de journaux des transactions est fixé par la valeur de l'option **innodb_log_files_in_group**. La taille des logs ib_logfile*, aussi appelés le **Redo log**, est fixée par l'option **innodb_log_file_size** mais ne peut pas dépasser 512 Go. Le buffer est aussi écrit

sur disque lors de chaque **commit** (validation) sauf si ce mécanisme a été modifié par la valeur de l'option **innodb_flush_log_at_trx_commit**.

Une partie des données ainsi que les index sont stockés en mémoire dans le **buffer pool**. La taille du buffer est fixée par la valeur de l'option **innodb_buffer_pool_size**. Il faut éviter que la taille de cette mémoire tampon dépasse les 80 % de la RAM du serveur. Chaque seconde, les informations dans le buffer sont écrites sur disque dans le tablespace. L'emplacement du tablespace est fixé par la valeur de l'option **innodb_data_file_path**.

Transactions

Le moteur Innodb est transactionnel. Ceci implique que le moteur collectionne des requêtes jusqu'à la commande **COMMIT**. A ce moment le moteur applique les modifications à toutes les tables concernées **en même temps**. Soit toutes les modifications réussissent, soit il y a un **ROLLBACK**. Par défaut, MySQL utilise un autocommit avec le moteur Innodb. Ceci implique qu'à la fin de chaque requête SQL, le serveur voit un commit virtuel.

Il est possible de modifier ce comportement automatique en utilisant la commande **BEGIN**. Par exemple :

```
mysql> BEGIN;
mysql> INSERT INTO table (colonne) VALUES (value);
mysql> COMMIT;
```

Si la dernière commande est remplacée par la commande ROLLBACK la modification apportée par la requête INSERT INTO sera effacée :

```
mysql> BEGIN;
mysql> INSERT INTO table (colonne) VALUES (value);
mysql> ROLLBACK;
```

Tablespace

Depuis la version 5.6.6 de MySQL, il existe l'option **innodb_file_per_table**. Si cette option est **0**, Innodb se comporte comme les versions précédentes, à savoir les données et les index de toutes les tables sont centralisées dans la **Tablespace**, c'est-à-dire, dans le fichier **ibdata** par défaut. Avec une valeur de **1**, les données et index de chaque table sont stockées dans un fichier individuel ayant une extension **.ibd** se trouvant dans le répertoire du schéma à côté du fichier de définition de la structure, le **.frm**.

Cette modification permet le stockage des fichiers .ibd sur un autre espace de stockage physique. De cette façon, il est possible d'utiliser des espaces de stockage très rapides.

Malgré l'utilisation des fichiers .ibd, il est toujours necéssaire d'avoir le fichier **ibdata** car ce dernier stocke le **dictionnaire des données** qui est une copie de tous les fichiers .frm ainsi que les Redo logs. Les Redo logs journalisent toute requête de modification des objets.

<note warning> Malgré l'utilisation des fichiers **.ibd**, il n'est pas possible de manipuler les fichiers dans un gestionnaire de fichiers. Ne les déplacez pas, ne les copiez pas sous peine de perdre des données. </note>

<note important> La quantité de mémoire libre dans un fichier *.ibd ou dans la Tablespace est indiquée par la sortie de la commande **SHOW TABLE STATUS**. Cette valeur est en pages et chaque page vaut 16 Ko. Bien que vous puissiez libérer de l'espace mémoire utilisée dans la Tablespace en passant la valeur de l'option **innodb_file_per_table** de **0** à **1** et en saisissant la commande **ALTER TABLE table ENGINE=InnoDB** pour chaque table, la taille du fichier ibdata ne diminuera pas. La seule façon de réduire la taille de ce fichier est d'exporter la base de données avec mysqldump puis de recréer la base de données à partir du dump. </note>

Multiversion Concurrency Control

Chaque table Innodb contient deux colonnes cachées :

- le numéro de transaction - **txn#**,
- un pointeur vers la version précédente de la ligne qui se trouve dans le Undo log. Ce pointeur s'appelle le **Pointeur ROLLBACK**.

Quand une ligne est modifiée, l'ancienne version de la ligne y compris son numéro de transaction est copiée dans le Undo log. La nouvelle version de la ligne y compris son numéro de transaction est copiée dans la Tablespace ou dans le fichier .ibd. Le pointeur de la nouvelle ligne identifie l'ancienne ligne dans le Undo log. Ce processus s'appelle le **MVCC** (*Multiversion Concurrency Control*).

Transaction Isolation Levels

Créez la base de données **nombres** :

```
mysql> CREATE DATABASE `Nombres` ;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> USE Nombres;
Database changed

mysql> CREATE TABLE english (id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, Number VARCHAR(10));
Query OK, 0 rows affected (0.08 sec)

mysql> INSERT INTO english (Number) VALUES ('One');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO english (Number) VALUES ('Two');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO english (Number) VALUES ('Three');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM english;
+----+--------+
| id | Number |
+----+--------+
|  1 | One    |
|  2 | Two    |
|  3 | Three  |
+----+--------+
3 rows in set (0.00 sec)

mysql>
```

Read Uncomitted

Mettez à jour l'enregistrement 1 sans faire de COMMIT ou de ROLLBACK :

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> UPDATE english SET Number = "Un" WHERE id = 1;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM english;
+----+--------+
| id | Number |
+----+--------+
|  1 | Un     |
|  2 | Two    |
|  3 | Three  |
+----+--------+
3 rows in set (0.00 sec)

mysql>
```

Ouvrez une autre terminal, connectez-vous à mysql et visualisez la modification :

```
[root@node01 ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.5.33-cll-lve MySQL Community Server (GPL) by Atomicorp

Copyright (c) 2000, 2013, 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.

mysql> USE Nombres;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed

mysql> SELECT * FROM english;
+----+--------+
| id | Number |
+----+--------+
|  1 | One    |
|  2 | Two    |
|  3 | Three  |
+----+--------+
3 rows in set (0.01 sec)

mysql>
```

<note important> Notez que le deuxième client voit l'ancienne valeur de l'enregistrement 1. Ceci est parce que la base de données a suivi le Pointeur ROLLBACK vers le Undo log pour afficher la valeur de la colonne. Notez que **MVCC** ne bloque pas la lecture malgré le fait qu'il y a une transaction en cours ! </note>

Pour voir la valeur dans la Tablespace, il faut modifier la valeur de l'option TRANSACTION ISOLATION LEVEL :

```
mysql> SET TRANSACTION ISOLATION LEVEL read uncommitted;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM english;
+----+--------+
| id | Number |
+----+--------+
|  1 | Un     |
|  2 | Two    |
|  3 | Three  |
+----+--------+
3 rows in set (0.01 sec)

mysql>
```

<note warning> Dans ce mode, le client voit une donnée fictive car il n'y a pas encore eu de COMMIT ! </note>

Read Comitted

Dans ce mode de fonctionnement InnoDB suit les Pointeurs ROLLBACK jusqu'à il trouve la dernière version ayant subi un COMMIT :

```
mysql> SET TRANSACTION ISOLATION LEVEL read committed;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM english;
+----+--------+
| id | Number |
+----+--------+
|  1 | One    |
|  2 | Two    |
|  3 | Three  |
+----+--------+
3 rows in set (0.01 sec)

mysql>
```

Retournez au premier terminal :

```
mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Retournez au deuxième terminal :

```
<mysql> SELECT * FROM english;
+----+--------+
```

```
| id | Number |
+----+--------+
|  1 | Un     |
|  2 | Two    |
|  3 | Three  |
+----+--------+
3 rows in set (0.00 sec)

mysql>
```

MyISAM

L'architecture du moteur MyISAM est plus simple que le moteur InnoDB. Les principaux caractéristiques du moteur MyISAM sont :

- Il n'est pas transactionnel,
- Il implémente un verrou au niveau table,
- Il n'y a pas de recouvrement automatique des données lors d'un crash,
- La structure de la table est stockée dans un fichier .frm,
- Les données sont stockées dans un fichier .MYD tandis que les index sont stockés dans un fichier .MYI,
- Il ne supporte pas de clefs étrangères,
- Il ne peut pas être sauvegardé à chaud,
- Il peut être compressé,
- Il supporte les index plain text (fulltext),
- Il est recommandé pour des applications avec beaucoup de lectures.

Il existe trois types de moteurs MyISAM : FIXED , DYNAMIC et COMPRESSED :

MyISAM FIXED

Une table est FIXED si elle ne contient aucun champ de type VARCHAR, VARBINARY, TEXT ou BLOB ou si la valeur de l'option **row_format** est FIXED (row_format=FIXED). Tous les enregistrements sont de la même taille. De ce fait, les tables sont plus rapides et plus robustes que les tables dynamiques mais elles prennent plus d'espace disque. Les données de la table ne peuvent pas être fragmentées. Une table devient FIXED quand la

valeur de l'option **row_format** est FIXED (row_format=FIXED) :

```
mysql> SHOW TABLE STATUS LIKE 'db' \G;
*************************** 1. row ***************************
           Name: db
         Engine: MyISAM
        Version: 10
     Row_format: Fixed
           Rows: 2
 Avg_row_length: 440
    Data_length: 880
Max_data_length: 123848989752688639
   Index_length: 5120
      Data_free: 0
 Auto_increment: NULL
    Create_time: 2014-01-22 15:28:03
    Update_time: 2014-01-22 15:28:04
     Check_time: 2014-01-22 15:28:04
      Collation: utf8_bin
       Checksum: NULL
 Create_options:
        Comment: Database privileges
1 row in set (0.01 sec)

ERROR:
No query specified

mysql>
```

MyISAM DYNAMIC

Une table est DYNAMIC si elle contient au moins un champ de type VARCHAR, VARBINARY, TEXT ou BLOB ou la valeur de l'option **row_format** est DYNAMIC (row_format=DYNAMIC). Les données de la table peuvent être fragmentées. Dans ce cas, l'utilisation de la commande OPTIMIZE TABLE ou

myisamchk est nécessaire :

```
mysql> SHOW TABLE STATUS LIKE 'proc' \G;
*************************** 1. row ***************************
           Name: proc
         Engine: MyISAM
        Version: 10
     Row_format: Dynamic
           Rows: 0
 Avg_row_length: 0
    Data_length: 0
Max_data_length: 281474976710655
   Index_length: 2048
      Data_free: 0
 Auto_increment: NULL
    Create_time: 2014-01-22 15:28:04
    Update_time: 2014-01-22 15:28:04
     Check_time: NULL
      Collation: utf8_general_ci
       Checksum: NULL
 Create_options:
        Comment: Stored Procedures
1 row in set (0.01 sec)

ERROR:
No query specified

mysql>
```

MyISAM COMPRESSED

Après avoir arrêté le serveur, la compression est obtenue en utilisant la commande **myisampack** suivi de la commande **myisamchk -rq**. Le décompression est obtenue en utilisant la commande **myisamchk –unpack**. La compression peut atteindre les 70% maximum. Une fois compressée

les requêtes UPDATE, DELETE et INSERT ne peuvent pas être utilisées :

```
[root@centos ~]# myisampack /var/lib/mysql/mysql/help_keyword.MYI
Compressing /var/lib/mysql/mysql/help_keyword.MYD: (467 records)
- Calculating statistics
- Compressing file
95.21%
Remember to run myisamchk -rq on compressed tables

[root@centos ~]# myisamchk -rq /var/lib/mysql/mysql/help_keyword.MYI
- check record delete-chain
- recovering (with sort) MyISAM-table '/var/lib/mysql/mysql/help_keyword.MYI'
Data records: 467
- Fixing index 1
- Fixing index 2

[root@centos ~]# myisamchk --unpack /var/lib/mysql/mysql/help_keyword.MYI
- recovering (with sort) MyISAM-table '/var/lib/mysql/mysql/help_keyword.MYI'
Data records: 467
- Fixing index 1
- Fixing index 2
```

Particularités

- Le moteur MyISAM est particulièrement portable car il suffit de copier les fichiers *.frm, *.MYD et *.MYI pour obtenir une sauvegarde physique,
- Le nombre d'enregistrements est stocké dans la table,
- Les index non-uniques peuvent être désactivés momentanément,
- Il est possible d'avoir des INSERT et SELECT simultanés en utilisant l'option **concurrent_insert**.

Memory

Les principaux caractéristiques du moteur Memory sont :

- Il n'est pas transactionnel,
- Il implémente un verrou au niveau table,
- Il n'y a pas de recouvrement automatique des données lors d'un crash,
- La structure de la table est stockée dans un fichier .frm,
- Les données sont stockées dans la mémoire,
- Il ne supporte pas de clefs étrangères,
- Il ne peut pas être sauvegardé à chaud,
- Il est recommandé pour des applications ayant besoin de la vitesse.

Particularités

- La consommation de la mémoire est fixée par les options **MAX_ROWS** et **max_heap_table_size**,
- Le moteur peut implémenter soit l'algorithme **HASH** pour les index pour augmenter la performance pour les recherches d'égalité soit l'algorithme **BTREE** pour les index pour augmenter la performance pour les recherches d'inégalité,
- Au démarrage du serveur les tables sont évidement vides. Il est possible de les pré-remplir grâce à des instructions SQL contenues dans un fichier référencé par l'option **init-file**.

En Pratique

Créez une table au format MEMORY en utilisant les données de la tables famille issue de la base carnetadresses :

```
mysql> CREATE TABLE tempfamilles TYPE=MEMORY SELECT * FROM familles ;
```

Visualisez les enregistrements de la table tempfamilles :

```
mysql> SELECT * FROM tempfamilles;
```

Supprimez la table tempfamilles :

```
mysql > DROP TABLE tempfamilles;
```

Archive

Les principaux caractéristiques du moteur Archive sont :

- Il n'est pas transactionnel,
- Il implémente un verrou au niveau enregistrement,
- Il n'y a pas de recouvrement automatique des données lors d'un crash,
- Les données sont stockées en les compressant,
- Il ne supporte pas de clefs étrangères,
- Il ne peut pas être sauvegardé à chaud,
- Il ne supporte pas d'Index,
- Il est recommandé pour réduire l'espace disque utilisé par les tables.

Particularités

- Le moteur ne permet que des requêtes INSERT et SELECT.

CSV

Les principaux caractéristiques du moteur CSV sont :

- Il n'est pas transactionnel,
- Il implémente un verrou au niveau table,
- Il n'y a pas de recouvrement automatique des données lors d'un crash,
- La structure de la table est stockée dans un fichier .frm,
- Les données sont séparées par des **;** et stockées dans un fichier .CSV tandis que les informations concernant l'état de la table ainsi que le nombre d'enregistrements sont stockées dans un fichier .CSM,
- Il ne supporte pas de clefs étrangères,
- Il ne peut pas être sauvegardé à chaud,
- Il ne supporte pas d'Index,
- Il est recommandé pour des tables ayant besoin d'être manipulées par des logiciels tiers.

FEDERATED

Les principaux caractéristiques du moteur FEDERATED sont :

- Il n'est pas transactionnel,
- Il n'y a pas de recouvrement automatique des données lors d'un crash,
- Il ne supporte pas de clefs étrangères,
- Il ne peut pas être sauvegardé à chaud,
- Il ne supporte pas d'Index,
- Il supporte des requêtes SELECT, INSERT, UPDATE, DELETE, TRUNCATE et DROP TABLE,
- Il n'utilise pas le cache de requêtes,
- Il est utilisé pour accéder à des données localisées sur un serveur **MySQL** distant.

NDB Cluster

Le moteur NDB Cluster est le moteur de MySQL Cluster. Les principaux caractéristiques du moteur NDB Cluster sont :

- Il est transactionnel de type **ACID** (Atomicité, Cohérence, Isolation, Durabilité),
- Il implémente le **MVCC** (Multi Version Concurrency Control) qui permet d'avoir des lectures qui ne bloquent pas des écritures et inversement,
- Il implémente un verrou au niveau enregistrement,
- Il implémente la réplication synchrone,
- Il implémente le basculement automatique sur un autre nœud en cas de panne et la synchronisation automatique du nœud à son démarrage,
- Il peut être sauvegardé à chaud,
- Il permet le dimensionnement des requêtes de lecture et d'écriture,
- Il utilise l'architecture **Shared Nothing** où les nœuds ne partagent pas un disque de données,
- Il implémente l’absence du **SPOF** (*Single Point Of Failure*).

Autres Moteurs Non Standards

XtraDB

XtraDB est le moteur de la société **Percona**. Il est basé sur InnoDB et a pour but d'être plus performant. Les principaux caractéristiques du moteur XtraDB sont :

- Il est transactionnel de type **ACID** (Atomicité, Cohérence, Isolation, Durabilité),
- Il implémente le **MVCC** (Multi Version Concurrency Control) qui permet d'avoir des lectures qui ne bloquent pas des écritures et inversement,
- Il implémente un verrou au niveau enregistrement,
- Il implémente un mécanisme de restauration automatique,
- Il supporte l'utilisation de clefs étrangères,
- Il peut être sauvegarder à chaud,
- Le moteur peut implémenter soit l'algorithme **HASH** pour les index pour augmenter la performance pour les recherches d'égalité soit l'algorithme **B+TREE** pour les index pour augmenter la performance pour les recherches d'inégalité.

Aria

Aria est le moteur de **Michael Widenius**. Il a été créé pour être un alternatif au moteur MyISAM. Les principaux caractéristiques du moteur Aria sont :

- En version 2.0, il est transactionnel de type **ACID** (Atomicité, Cohérence, Isolation, Durabilité),
- Il implémente un verrou au niveau enregistrement,
- Il implémente un mécanisme de restauration automatique.

Jointures

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

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

FULL JOIN

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

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

ou

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

LEFT JOIN

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

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

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

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

RIGHT JOIN

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

LAB #2 - Le Langage SQL

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

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

mysql>
```

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

```
mysql> USE CarnetAdresses ;
Database changed
mysql>
```

```
mysql> CREATE TABLE familles (CodeFamille BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY, Nom VARCHAR(40),
    -> PrenomPere VARCHAR(40) , PrenomMere VARCHAR(40) , Adresse1 VARCHAR(40) , Adresse2 VARCHAR(40),
    -> CodePostal VARCHAR(5), Ville VARCHAR(40), ProfPere VARCHAR(40), ProfMere VARCHAR(40));
Query OK, 0 rows affected (0.03 sec)

mysql>
```

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

familles

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

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

mysql>
```

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

enfants

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

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

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| CarnetAdresses     |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

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

mysql>
```

```
mysql> SHOW COLUMNS FROM familles FROM CarnetAdresses;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| CodeFamille | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| Nom         | varchar(40) | YES  |     | NULL    |                |
| PrenomPere  | varchar(40) | YES  |     | NULL    |                |
| PrenomMere  | varchar(40) | YES  |     | NULL    |                |
| Adresse1    | varchar(40) | YES  |     | NULL    |                |
| Adresse2    | varchar(40) | YES  |     | NULL    |                |
| CodePostal  | varchar(5)  | YES  |     | NULL    |                |
| Ville       | varchar(40) | YES  |     | NULL    |                |
| ProfPere    | varchar(40) | YES  |     | NULL    |                |
| ProfMere    | varchar(40) | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
10 rows in set (0.00 sec)

mysql>
```

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

mysql>
```

```
mysql> SHOW TABLE STATUS FROM CarnetAdresses;
+----------+--------+---------+------------+------+----------------+-------------+-----------------+------------
-+-----------+----------------+---------------------+-------------+-----------+-------------------+----------+--
-------------+---------+
| Name     | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length
| Data_free | Auto_increment | Create_time         | Update_time | Check_time | Collation         | Checksum |
Create_options | Comment |
+----------+--------+---------+------------+------+----------------+-------------+-----------------+------------
-+-----------+----------------+---------------------+-------------+-----------+-------------------+----------+--
-------------+---------+
| Enfants  | InnoDB |      10 | Compact    |    0 |              0 |       16384 |               0 |           0
|   9437184 |           NULL | 2014-01-27 10:46:10 | NULL        | NULL      | latin1_swedish_ci |     NULL |
|         |
| familles | InnoDB |      10 | Compact    |    0 |              0 |       16384 |               0 |           0
|   9437184 |              1 | 2014-01-27 10:45:39 | NULL        | NULL      | latin1_swedish_ci |     NULL |
|         |
+----------+--------+---------+------------+------+----------------+-------------+-----------------+------------
-+-----------+----------------+---------------------+-------------+-----------+-------------------+----------+--
-------------+---------+
2 rows in set (0.00 sec)

mysql>
```

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

data

```
INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Durant', 'Jacques', 'Jane', '23 rue Dutor','','92200', 'Neuilly', 'Plombier',''); INSERT
INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Tortua', 'Benoît', 'Carole', '7 rue Verget','', '75005', 'Paris', 'Cadre', 'Cadre');
INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Portier', 'Pierre', 'Elisabeth', '5 rue Toulet','', '92200', 'Neuilly', 'Dentiste','');
INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
```

```
ProfMere) VALUES ('Renault', 'Damien', 'Anne', '2 rue Ragon','', '75007', 'Paris', 'Comptable',
'Enseignante'); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal ,
Ville , ProfPere , ProfMere) VALUES ('Darduet', 'Jean','','','','','','',''); INSERT INTO familles (Nom ,
PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES ('Rodier',
'Gérard', 'Aurélie', '6 rue Agien', '77000', 'Fontainebleau', 'Professeur',''); INSERT INTO familles (Nom ,
PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES ('Tarte',
'Alla', 'Crème', '1 allée Durond','', '92200', 'Neuilly', 'Cuisinier', 'Cuisinière'); INSERT INTO familles
(Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere , ProfMere) VALUES
('Cohen', 'David', 'Sarah', '7 Av d\'Eylau','', '75016', 'Paris', 'PDG', 'Assistante Direction'); INSERT
INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Dupont2', 'Bruno', 'Odile', '12 rue Sébastien','', '75008', 'PARIS', 'Electricien',
'Comptable'); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville
, ProfPere , ProfMere) VALUES ('Durand2', 'Jacques', 'Jane', '23 rue Dutor','', '92200', 'Neuilly',
'Plombier',''); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal ,
Ville , ProfPere , ProfMere) VALUES ('Darmon2', 'Bruno', 'Béatrice', '2 rue Nicolo','', '13008',
'Marseille', 'Cadre', 'Peintre'); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2
, CodePostal , Ville , ProfPere , ProfMere) VALUES ('Darduet2', 'Jean', '','' ,'' ,'' ,'','','',''); INSERT
INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal , Ville , ProfPere ,
ProfMere) VALUES ('Tarte2', 'Alla', 'Crème', '1 allée Durond','', '92200', 'Neuilly', 'Cuisinier',
'Cuisinière'); INSERT INTO familles (Nom , PrenomPere , PrenomMere , Adresse1 , Adresse2 , CodePostal ,
Ville , ProfPere , ProfMere) VALUES ('Cohen2', 'David', 'Sarah', '7 Av d\'Eylau','', '75016', 'Paris',
'PDG', 'Assistante Direction');
```

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

1	ALTER TABLE Enfants ADD INDEX (Prenom);
2	ALTER TABLE Enfants DROP INDEX Prenom;
3	ALTER TABLE Enfants ADD TelPort VARCHAR(50) NOT NULL;
4	ALTER TABLE Enfants CHANGE TelPort TelPort VARCHAR(22) NOT NULL;
5	ALTER TABLE Enfants ADD Test VARCHAR(40) NOT NULL;
6	ALTER TABLE Enfants DROP Test;

```
7 SELECT * FROM familles;
8 SELECT * FROM familles LIMIT 8,12;
```

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

```
9 UPDATE familles SET Adresse1 = '120 rue de Vaugirard', CodePostal = '75015', Ville = 'PARIS' WHERE Nom = 'DARDUET' AND PrenomPere = 'Jean' LIMIT 1;
10 DELETE FROM familles WHERE Nom = 'DURANT' AND PrenomPere = 'Jacques';
11 DELETE FROM familles WHERE Ville = 'Neuilly';
12 SELECT * FROM familles LIMIT 8,12;
13 SELECT nom, PrenomPere FROM familles;
14 SELECT Nom AS 'Nom de famille', PrenomPere AS 'Prenom du Père' FROM familles;
15 INSERT INTO Enfants (Prenom, Sexe, CodeFamille) VALUES ('Georges', 'M', '14');
16 SELECT familles.nom, familles.PrenomPere FROM familles, Enfants;
```

<note> Créez la colonne **familles.nb_enfants**. </note>

```
17 UPDATE familles SET nb_enfants = '2' WHERE CodeFamille = '2';
18 UPDATE familles SET nb_enfants = '1' WHERE CodeFamille = '4';
19 UPDATE familles SET nb_enfants = '3' WHERE CodeFamille = '5';
20 UPDATE familles SET nb_enfants = '2' WHERE CodeFamille = '6';
21 UPDATE familles SET nb_enfants = '4' WHERE CodeFamille = '8';
22 UPDATE familles SET nb_enfants = '5' WHERE CodeFamille = '9';
23 UPDATE familles SET nb_enfants = '3' WHERE CodeFamille = '11';
24 UPDATE familles SET nb_enfants = '1' WHERE CodeFamille = '12';
25 UPDATE familles SET nb_enfants = '5' WHERE CodeFamille = '14';
26 SELECT SUM(nb_enfants) FROM familles;
27 SELECT MIN(nb_enfants) AS 'Nb enfants minimum', MAX(nb_enfants) AS 'Nb enfants maximum', AVG(nb_enfants) AS 'Nb enfants moyen' FROM familles;
28 SELECT nom FROM familles WHERE ville = 'Paris';
```

```
29 SELECT nom FROM familles WHERE ville = 'Paris' OR ville = 'Neuilly';
30 ALTER TABLE Enfants DROP PRIMARY KEY;
31 INSERT INTO Enfants (Prenom, Sexe, CodeFamille) VALUES ('Alex', 'F', '12');
32 INSERT INTO Enfants (Prenom, Sexe, CodeFamille) VALUES ('Mila', 'F', '2');
33 INSERT INTO Enfants (Prenom, Sexe, CodeFamille) VALUES ('Amandine', 'F', '5');
34 SELECT familles.Nom , Enfants.Prenom FROM familles, Enfants WHERE ( Enfants.CodeFamille = familles.CodeFamille );
35 SELECT familles.nom, familles.PrenomPere FROM familles, Enfants;
36 SELECT ville, min(nb_enfants) as 'Nb enfants minimum', max(nb_enfants) as ' Nb enfants maximum', avg(nb_enfants) as 'Nb enfants moyen' from familles GROUP BY Ville;
37 SELECT * FROM familles ORDER BY nom;
38 SELECT * FROM familles GROUP BY ville, Nom;
39 UPDATE familles SET Adresse2 = '-';
40 UPDATE familles SET Adresse2 = '___',nb_enfants=2;
41 UPDATE familles SET nb_enfants=4 WHERE Ville = 'Paris';
42 UPDATE familles SET nb_enfants=7 LIMIT 4;
43 UPDATE familles SET nb_enfants=8 ORDER BY Nom LIMIT 3;
44 DELETE FROM familles WHERE Nom = 'Dupont';
45 CREATE TABLE loisirs( Nom VARCHAR( 30 ) NOT NULL );
46 DROP TABLE loisirs;
47 INSERT INTO familles (Nom, PrenomPere) VALUES ('Alouet','Jean'),('Rahtmi','Patrick');
48 INSERT INTO familles (PrenomPere) SELECT Enfants.prenom FROM Enfants;
49 ALTER TABLE familles ADD Sports VARCHAR(50) NOT NULL;
50 ALTER TABLE familles ADD INDEX (Sports);
51 ALTER TABLE familles ADD INDEX (PrenomPere), ADD EmailPere VARCHAR( 50 ) NOT NULL;
52 ALTER TABLE familles CHANGE PrenomPere Prenom_Pere VARCHAR( 45 ) NOT NULL;
53 ALTER TABLE familles DROP Sports;
54 ALTER TABLE familles DROP INDEX PrenomPere;
55 ALTER TABLE familles DISABLE KEYS;
56 ALTER TABLE Enfants RENAME familles_enfants;
57 ALTER TABLE familles ORDER BY Prenom_Pere;
58 ALTER TABLE familles ADD Commentaire LONGTEXT NOT NULL ;
```

```
59 ALTER TABLE familles ADD FULLTEXT (Commentaire);
60 UPDATE familles SET Commentaire = 'La vie est un long fleuve tranquille' WHERE CodeFamille = '11';
61 UPDATE familles SET Commentaire = 'Paris, capitale de la France est traversée par un fleuve' WHERE CodeFamille ='4';
62 UPDATE familles SET Commentaire = 'Le ruisseau se jette dans la rivière qui se jette dans le FLEUVE' WHERE CodeFamille = '6';
63 SELECT * FROM familles WHERE MATCH (Commentaire) AGAINST ('fleuve');
64 SELECT Nom, Commentaire FROM familles WHERE MATCH (Commentaire) AGAINST ('-capitale+fleuve'IN BOOLEAN MODE);
```

<html>

</html>
