# Exercice №11
# Les fonctions de dates

Jean-Pierre Duchesneau, DFC Cégep Ste-Foy Automne 2018

**Évaluation :** formative
**Travail de préférence individuel.**
**Durée :** 2 heures
**Environnement :** MySQL Serveur 5.7
**Fichier SQL :** REMPARTS.CSV

## 1 Mise en contexte

"L'open data ou donnée ouverte est une donnée numérique dont l'accès et l'usage sont laissés libres aux usagers. ... L'ouverture des données (open data) est à la fois un mouvement, une philosophie d'accès à l'information et une pratique de publication de données librement accessibles et exploitables[...] Elle s'inscrit dans une tendance qui considère l'information publique comme un bien commun" [1]

Dans cet exercice nous allons utiliser des données ouvertes de la ville de Québec. Celle-ci sont généralement disponible dans les formats suivants : csv, xls, json, parfois d'autre format.

Ici nous allons utiliser les données du "Calendrier des rencontres des Remparts de Québec" de l'année 2012-13, elles sont fournies à l'adresse suivante `http://donnees.ville.quebec.qc.ca/catalogue.aspx`
. Procurez-vous le fichier **REMPARTS.CSV**

À titre d'information, il est possible d'avoir une liste des principales données ouvertes au Québec sur le site Web **Données Québec** à l'adresse : [`https://www.donneesquebec.ca/fr/`

## 2 Importer des données CSV

"Un fichier CSV est un fichier texte, par opposition aux formats dits « binaires ». Chaque ligne du texte correspond à une ligne du tableau et les virgules correspondent aux séparations entre les colonnes. Les portions de texte séparées par une virgule correspondent ainsi aux contenus des cellules du tableau.

1. `https://fr.wikipedia.org/wiki/Open_data`

Une ligne est une suite ordonnée de caractères terminée par un caractère de fin de ligne (line break – LF ou CRLF), la dernière ligne pouvant en être exemptée."[2] Il est aussi possible que les données soient séparées par d'autres types de caractères. C'est le cas dans notre fichier REMPARTS.CSV ou le caractère | (Pipe) a été utilisé.

## 2.1 Création de la structure d'accueil

D'abord nous allons créer une table qui pourra contenir les données.

```
1 Drop database if exists rempart;
2 create database rempart;
3
4 use rempart;
5 create table partie(
6 idPartie int primary key auto_increment,
7 datePartie datetime not null default '0000-00-00 00:00:00',
8 visiteur varchar(25) not null default 'null',
9 local varchar(25) not null default 'null');
```

Cette table contient tous les champs existants dans le fichier REMPARTS.CSV, plus une clé primaire auto-incrémentée.

## 2.2 Syntaxe de l'importation d'un fichier

La syntaxe expliquée, dans le chapitre 13.2.6 LOAD DATA INFILE Syntax de la documentation de MySQL, est la suivante :

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var
        [, col_name_or_user_var] ...)]
    [SET co_name={expr | DEFAULT},
        [, col_name={expr | DEFAULT}] ...]
```

2. https://fr.wikipedia.org/wiki/Comma-separated_values

Allons-y seulement avec les instructions nécessaires à notre cas :

```
LOAD DATA LOCAL INFILE '[Chemin absolu du fichier]\REMPARTS.CSV'
INTO TABLE partie
COLUMNS TERMINATED BY '|'
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(datePartie, visiteur, local);-- Il est important de préciser les champs importés.
```

À l'exécution de votre requête, vous devez avoir l'erreur suivante :

ERROR 1290 (HY000) : The MySQL server is running with the –secure-file-priv option so it cannot execute this statement

Pour des raisons de sécurité, il est impossible par défaut d'exécuter des fichiers sauf si on modifie le fichier de configuration du serveur ou encore qu'on place le fichier à l'endroit prévu sur le serveur. A noter qu'en mode production, cet emplacement n'est pas accessible de l'extérieure sauf par SSH.

Le fichier de configuration du serveur est situé à cet emplacement :

— En Windows `C:\ProgramData\MySQL\MySQLServer5.7\mi.ini`

— En Linux `/etc/mysql/my.cnf`

Il faut modifier la variable d'environnement **secure-file-priv** en fonction de nos besoins. La documentation de MySQL [3] nous donnes les informations suivantes concernant cette variable d'environnent :

- `secure_file_priv`

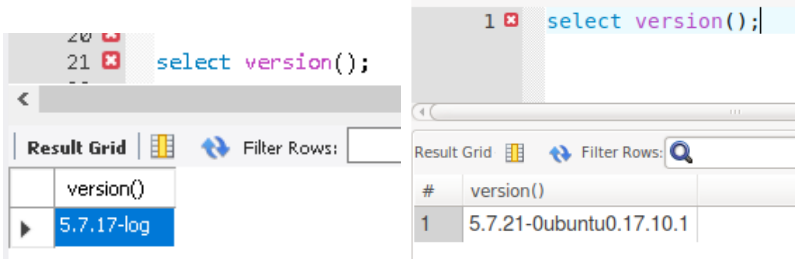
| Property | Value |
|---|---|
| **Command-Line Format** | `--secure-file-priv=dir_name` |
| **System Variable** | `secure_file_priv` |
| **Scope** | Global |
| **Dynamic** | No |
| **Type** | string |
| **Default** (>= 5.7.6) | `platform specific` |
| **Default** (<= 5.7.5) | `empty string` |
| **Valid Values** (>= 5.7.6) | `empty string`<br>`dirname`<br>`NULL` |
| **Valid Values** (<= 5.7.5) | `empty string`<br>`dirname` |

3. https ://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_secure_file_priv

Donc première chose, vérifiez la version de votre serveur avec la requête avec le code SQL suivant :

```sql
select version();
```

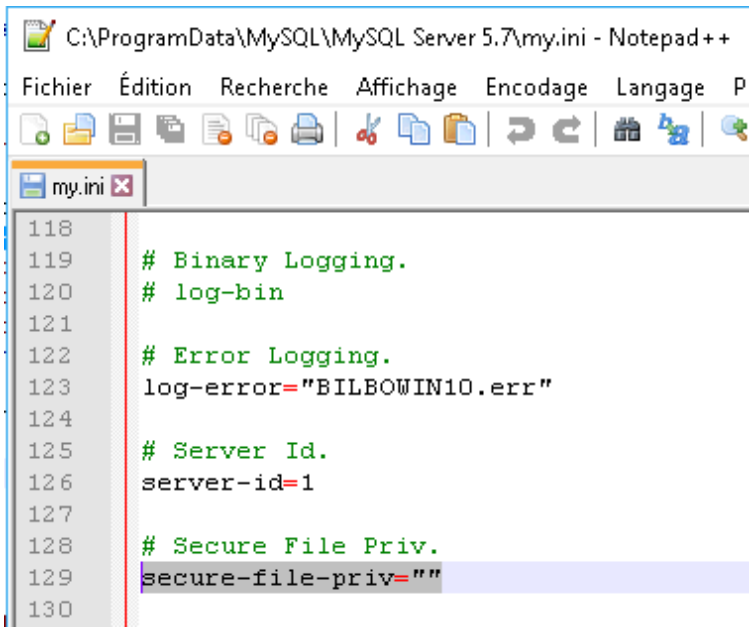
Voici les résultats sur Windows et sur Linux (VM) :




Ici on remarque qu'avec la version du serveur $>=$ à 5.7.6, ce qui est mon cas, il existe trois possibilités :

1) Met une chaine vide et permettre l'importation des données ;
2) Imposer un répertoire que nous considérons sécuritaire pour l'importation des données ;
3) Mette NULL, ce qui empêche l'importation des données.

Comme nous ne sommes pas en mode de production, nous allons opter pour la facilité, mettre la chaine vide.




Après modification du fichier, il est nécessaire de redémarrer le serveur MySQL :

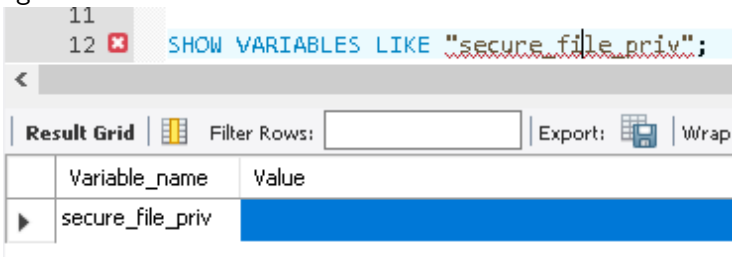
— En Windows :

1) Tapez **services** dans la zone de rechercher de Windows
2) Ouvrez la fenêtre des services et trouvez **MySQL57**.
3) Avec votre bouton droit de la souris (Menu contextuel) cliquez sur **Redémarrer**.

— En Linux :

1)

Par la suite, taper la commande SQL suivante pour vérifier la prise en compte de votre nouvelle configuration :




La variable doit être vide.

Tous est en place pour exécuter à nouveau la commande suivante :

```
1  LOAD DATA LOCAL INFILE '[Chemin absolu du fichier]\REMPARTS.CSV'
2  INTO TABLE partie
3  COLUMNS TERMINATED BY '|'
4  OPTIONALLY ENCLOSED BY '"'
5  ESCAPED BY '"'
6  LINES TERMINATED BY '\n'
7  IGNORE 1 LINES
8  (datePartie,visiteur, local);-- Il est important de préciser les champs importés.
```

Vous devriez avoir le message suivant :

```
68 row(s) affected Records : 68 Deleted : 0 Skipped : 0 Warnings : 0
```

Faite un select pour vérifier votre table :

```
1  select * from partie;
```

# 3 Les fonctions dates

Répondez aux questions suivantes. Les réponses vous sont fournies à la fin de l'exercice.

1) Donner le nombre de parties en janvier ?
2) Combien de fois jouent les Remparts un deuxième jour du mois ?
3) Sélectionnez les parties qui ont lieu dans les 8 premières semaines de l'année 2013.
4) Afficher le jour (en chiffres) et le mois (en toutes lettres), les visiteurs et les locaux, pour les parties ayant lieu à l'extérieur dans les deux dernières semaines de l'année 2012.
5) Même chose que la question précédente, mais placer la date dans un seul champ avec jour en lettre, date en chiffre, mois en lettre et années en chiffre. Ex('Sunday 30 December 2012')
6) Afficher la première partie, la dernière partie et calculer la différence (nombre de jour) entre la première et la dernière partie.

7) Afficher la date des parties dans le format suivant : Frideay, 12 dec 2013 à 07 :00 PM.

## 3.1 Les fonctions dates : réponses

1) Donner le nombre de parties en janvier ?

```
1    select count(*) from partie where month(datePartie)=1;
```

2) Combien de fois jouent les Remparts un deuxième jour du mois ?

```
1    select count(*) from partie where day(datePartie)=2;
```

3) Sélectionnez les parties qui ont lieu dans les 8 premières semaines de l'année 2013.

```
1    select * from partie where weekofyear(datePartie)<9;
```

4) Afficher le jour (en chiffres) et le mois (en toutes lettres), les visiteurs et les locaux, pour les parties ayant lieu à l'extérieur dans les deux dernières semaines de l'année 2012.

```
1    select datePartie, day(datePartie) as Jour, monthname(datePartie) as mois, ↩
         visiteur, local
2    from partie
3    where Week(datePartie) >50;
```

5) Même chose que la question précédente, mais placer la date dans un seul champ avec jour en lettre, date en chiffre, mois en lettre et années en chiffre. Ex('Sunday 30 December 2012')

```
1    select datePartie, date_format(datePartie, '%W %d %M %Y') as Jour, monthname(↩
         datePartie) as mois, visiteur, local
2    from partie
3    where Week(datePartie) >50;
```

6) Afficher la première partie, la dernière partie et calculer la différence (nombre de jour) entre la première et la dernière partie.

```
1    select min(datePartie) as PremièrePartie, max(datePartie) As DernièrePartie,
2        datediff(max(datePartie), min(datePartie)) as nbJour
3    from partie;
```

7) Afficher la date des parties dans le format suivant : Frideay, 12 dec 2013 à 07 :00 PM.

```
1    select date_format(datePartie, '%W, %d %b %Y à %l:%i %p ') as Date
2    from partie;
```

# 4 Jouons avec les dates

Rien de mieux que des tables de test sur les dates : Créez la structure suivante :

```
1  -- Exercice 11 date et heures
2  Drop database if exists exercice11;
3  create database exercice11;
4  use exercice11;
5
6  create table tbl_test_date (
7  id_date  INT primary key auto_increment,
8  td_date  DATE not null default '0000-00-00',
9  td_heure time not null default '00:00:00',
10 td_date_time   datetime not null default '0000-00-00 00:00:00',
11 td_insertion_date Timestamp not null default current_timestamp,
12 td_maj_date timestamp default current_timestamp on update current_timestamp,
13 td_note text null
14 );
```

Analysons on peut la structure :

— Il est important d'utiliser les valeurs par défauts pour ne pas avoir d'erreurs dans nos calculs par la suite (lignes 8, 9 et 10).

— Les fonctions peuvent être utilisées dans des valeurs par défauts (lignes 11 et 12)

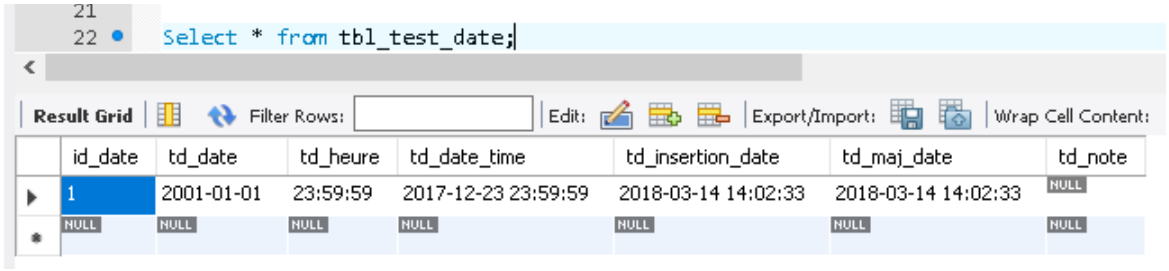
saisissons l'enregistrement suivant :

```
1 INSERT INTO tbl_test_date (id_date, td_date, td_heure, td_date_time)
2 values (1, '2001-01-01','23:59:59','2017-12-23 23:59:59');
```

Vérifions le résultat :

```
1 Select * from tbl_test_date;
```

Résultat :




— Nos deux fonctions ont saisi les informations dans les champs td_insertion_date et td_maj_date timestamp.

Saisissons è nouveau un enregistrement :

```
INSERT INTO tbl_test_date (td_note)
values ('Utilisation des valeurs par défaut');
```

Vérifiez le résultat, que remarqué vous ?

Réponse : Tous les champs, sauf un, ont été renseignés par les valeurs par défauts.

| id_date | td_date | td_heure | td_date_time | td_insertion_date | td_maj_date | td_note |
|---|---|---|---|---|---|---|
| 1 | 2001-01-01 | 23:59:59 | 2017-12-23 23:59:59 | 2018-03-14 14:02:33 | 2018-03-14 14:02:33 | NULL |
| 2 | 0000-00-00 | 00:00:00 | 0000-00-00 00:00:00 | 2018-03-14 14:12:37 | 2018-03-14 14:12:37 | Utilisation des valeurs par défaut |

Maintenant voyons ce qui arrivé lors de la modification des données. Exécutez la requête suivante :

```
update tbl_test_date set TD_DATE ='2018-01-01' where id_date = 2;
Select * from tbl_test_date;
```

— Remarquez la date de mise à jour a changé et non la date d'insertion. C'est le rôle de la fonction **current_timestamp on update current_timestamp** dans la construction de la table.

| id_date | td_date | td_heure | td_date_time | td_insertion_date | td_maj_date | td_note |
|---|---|---|---|---|---|---|
| 1 | 2001-01-01 | 23:59:59 | 2017-12-23 23:59:59 | 2018-03-14 14:02:33 | 2018-03-14 14:02:33 | NULL |
| 2 | 2018-01-01 | 00:00:00 | 0000-00-00 00:00:00 | 2018-03-14 14:12:37 | 2018-03-14 14:18:42 | Utilisation des valeurs par défaut |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Cette fois, insérons les enregistrements suivants :

```
INSERT INTO tbl_test_date
values (3, '2001-02-01','23:59:59','2017-12-24 23:59:59',null, null, 'Insertion suplé↩
    mentaire avec null');

INSERT INTO tbl_test_date
values (4, '2001-02-01','23:59:59','2017-12-24 23:59:59',null, null, 'Insertion suplé↩
    mentaire ');

INSERT INTO tbl_test_date
values (5, curdate(), curtime(),now(),null, null, 'Insertion suplémentaire avec fonction')↩
    ;
```

Votre table devrait ressembler à ceci :

| id_date | td_date | td_heure | td_date_time | td_insertion_date | td_maj_date | td_note |
|---|---|---|---|---|---|---|
| 1 | 2001-01-01 | 23:59:59 | 2017-12-23 23:59:59 | 2018-03-14 14:02:33 | 2018-03-14 14:02:33 | NULL |
| 2 | 2018-01-01 | 00:00:00 | 0000-00-00 00:00:00 | 2018-03-14 14:12:37 | 2018-03-14 14:18:42 | Utilisation des valeurs par défaut |
| 3 | 2001-02-01 | 23:59:59 | 2017-12-24 23:59:59 | 2018-03-14 14:24:23 | 2018-03-14 14:24:23 | Insertion suplémentaire avec null |
| 4 | 2001-02-01 | 23:59:59 | 2017-12-24 23:59:59 | 2018-03-14 14:24:29 | 2018-03-14 14:24:29 | Insertion suplémentaire |
| 5 | 2018-03-14 | 14:24:29 | 2018-03-14 14:24:29 | 2018-03-14 14:24:29 | 2018-03-14 14:24:29 | Insertion suplémentaire avec fonction |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

# 5 Exercice facultatif

Si vous sentez le besoin d'approfondir les fonctions dates, vous pouvez faire le chapitre 27 du livre de Chantal Gribaumont [4]

**Fin de l'exercice 11.**

4. Gribaumont, Chantal, "Administrez vous bases de données avec MySQL 2ième édition." Open Classrooms. Disponible sur LÉA.

# Sommaire



---

***Ce document a été écrit avec LaTeX.***