

Cette page traitera principalement du système de gestion de base de données (SGBD) MySQL et, sous l'angle de celui-ci, de SQL dans le cadre de l'application EzGED.

# Créer une base de données

Pour créer une base de données nommée *nchp\_test*

```
mysql -u root -p -e "create database nchp_test;"
```

## Sauvegarder / Exporter

Pour sauvegarder notre base de données nous allons réaliser un dump.

### Structures et données dans un seul fichier

Ouvrez un interpréteur de commandes et entrez la commande suivante :

```
mysqldump -u root -p nom_base > dump_base.sql
```

Si le répertoire des binaires de mysql n'est pas dans les variables d'environnement il faudra indiquer le chemin complet vers la commande. Par exemple:

```
C:\nchp\mysql\bin\mysqldump -u root -p nom_base > dump_base.sql
```

Où *nom\_base* doit être remplacé par le nom de votre base de données (pour EzGED par défaut ce nom est **nchp\_ezged**)  
et *dump\_base.sql* est le nom du fichier de sauvegarde (vous pourriez l'appeler autrement bien entendu).

```
mysqldump -u root -p nom_base > dump_base.sql
```

### Structures et données dans des fichiers séparés

Pour exporter la structure (triggers et procédures comprises) sans les données entrez la commande suivante :

```
mysqldump -u root -p --routines --no-data nom_base > dump_structure_base.sql
```

Pour exporter les données seules entrez la commande suivante:

```
mysqldump -u root -p --no-create-info --skip-triggers nom_base >  
dump_datas_base.sql
```

## Sélectionner des tables à dumper

Il est possible d'exporter uniquement certaines tables avec cette commande :

```
mysqldump -u root -p base table_1 table_2 ... table_n >  
dump_selection_tables.sql
```

## Reparer les tables InnoDB

Si besoin de réparer les tables InnoDB vous pouvez utiliser la fonction `db_regenerate_innodb_indexes()` du module python `db_mysql` livré par EzGED.

Pour les instructions voir la page [de documentation du module db\\_mysql](#)

## Importer / Restaurer

### Structures et données dans un seul fichier

```
mysql -u root -p ma_base < dump_base.sql
```

### Structures et données dans des fichiers séparés

On remonte d'abord la structure:

```
mysql -u root -p ma_base < dump_structure_base.sql
```

Puis on remonte les données:

```
mysql -u root -p ma_base < dump_datas_base.sql
```

## Réparation de tables

Réparation d'une table dans phpMyadmin ou dans l'invite mysql (bref un client SQL connecté à la base de données):

```
REPAIR TABLE <nom_de_la_table>
```

Réparation auto de toutes les tables de toutes les bases de données :

```
mysqlcheck -u root -p --auto-repair --check --optimize --all-databases
```

Si la table est en myisam et que la réparation ne fonctionne pas vous pouvez tenter ceci :

```
myisamchk --safe-recover chemin/fichier.MYI
```

## Remarques

Si pour l'une des commandes ci-dessus l'interpréteur vous signifie que la commande n'est pas reconnue, deux solutions possibles.

1/ Déplacez vous jusqu'au niveau du répertoire contenant les binaires mysql :

```
cd C:\nchp\mysql\bin
```

ou

2/ Assurez vous que le chemin ci-dessus soit bien présent dans la variable **path** des variables d'environnement windows.

La plupart des opérations ci-dessus sont accessibles via l'interface graphique phpmyadmin mais la ligne de commande est plus performante. Avec phpmyadmin vous seriez par exemple limité sur la taille du fichier sql pour l'import en fonction des paramètres de configuration de php.

## Base de données d'archivage

Objectif: déplacer sur une autre base de données des tables qui sont en consultation afin d'alléger les dumps.

La base de données d'archive reste sur la même instance il n'y aura donc aucune difficulté pour continuer à y accéder depuis la ged.

On commence par créer la base de données

```
CREATE DATABASE nchp_archive;
```

La procédure est ensuite la suivante ([Documentation MariADB](#)):

```
FLUSH TABLES db_name.table_name FOR EXPORT
```

```
# Copy the relevant files associated WITH the TABLE
```

```
UNLOCK TABLES;
```

# Migration vers MySQL 5.6

Il est nécessaire d'ajouter le paramètre de configuration suivant dans le .ini:

```
[mysqld]
secure_auth=0
```

Par défaut dans MySQL 5.6 `secure_auth` est à 1 et bloque toute tentative d'authentification avec un mot de passe haché avec l'algorithme pré-4.1

[https://dev.mysql.com/doc/refman/5.0/en/server-options.html#option\\_mysqld\\_secure-auth](https://dev.mysql.com/doc/refman/5.0/en/server-options.html#option_mysqld_secure-auth)

<https://dev.mysql.com/doc/refman/5.0/en/old-client.html>

## Réinitialiser le mot de passe root

Suivre la procédure indiquée sur la documentation de MySQL :

<https://dev.mysql.com/doc/mysql-windows-excerpt/5.7/en/resetting-permissions-windows.html>

## Configuration adaptée aux tables volumineuses

Voici une configuration adaptée aux instances MySQL/MariaDB traitant des volumes de données importants.

```
[mysqld]
datadir=C:/nchp/mysql/data
port=3306
#EGE-20190429 : Ajout-Modif suite retour Fabrice
#innodb_buffer_pool_size=1023M
sql_mode="STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION"
default_storage_engine=innodb
skip-external-locking
key_buffer_size = 512M
table_open_cache = 512
sort_buffer_size = 2M
read_buffer_size = 2M
read_rnd_buffer_size = 8M
myisam_sort_buffer_size = 64M
thread_cache_size = 8
query_cache_size = 64M
thread_concurrency = 10
max_connections = 1000
#EGE-20201230 : Optimisation suite erreurs bdd
```

```
innodb_buffer_pool_size= 4G
wait_timeout = 43200
max_allowed_packet = 64M
innodb_log_file_size= 20M

#EGE-20190429 : Traces requetes longues
slow_query_log=1
slow_query_log_file=C:/nchp/mysql/log/mysql-slow.log
long_query_time = 2
net_read_timeout = 60
connect_timeout = 20

[client]
port=3306
plugin-dir=C:/nchp/mysql/lib/plugin
```

From:

<https://wiki.ezdev.fr/> - **EzGED Wiki**

Permanent link:

<https://wiki.ezdev.fr/doku.php?id=mysql&rev=1708095238>



Last update: **2024/02/16 14:53**