

Procédure d'exportation et d'importation d'une base de données

Nous allons procéder à l'exportation de notre base de données suite à changement de site ou juste migration en local .

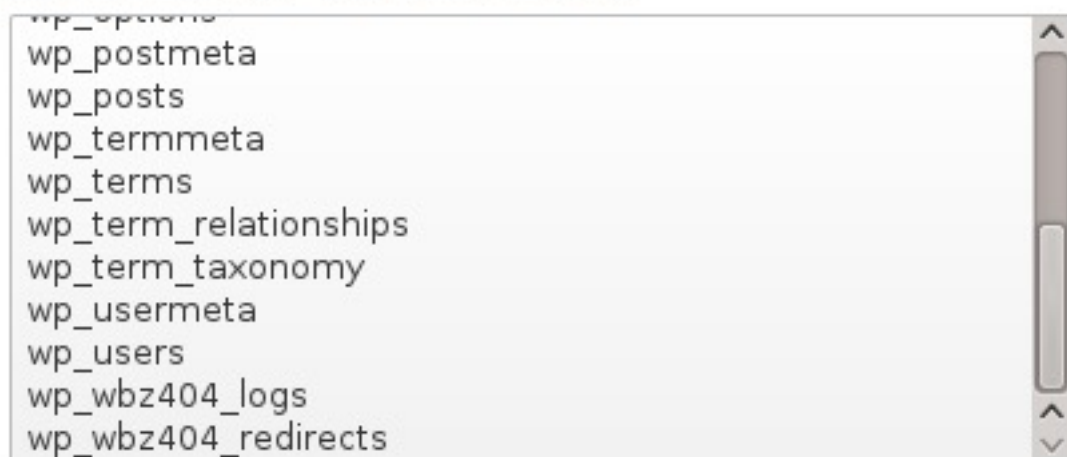
Rentrer dans phpMyadmin via votre fournisseur ou en localhost, sélectionnez ensuite l'onglet exporter et choisissez l'exportation personnalisée :

Méthode d'exportation :

- Rapide - n'afficher qu'un minimum d'options
- Personnalisée - afficher toutes les options possibles

Table(s) :

Tout sélectionner / Tout désélectionner



Sélectionnez toute la base.

Laisser les sorties comme elles sont :

Sortie :

Renommer les bases de données / tables / colonnes exportées

Diriger la sortie vers un fichier

Modèle de nom de fichier : utiliser ceci pour les futures exportations

Jeu de caractères du fichier :

Compression :

Afficher les résultats

Ignorer les tables de taille plus grande que Mio

Choisir le format de sortie de votre base de données, dans mon cas en .sql :

Format :

Dans les options spécifiques au format cocher structure et données :

Options spécifiques au format :

Afficher les commentaires (incluant les informations telles que l'horodatage d'exportation, la version de PHP et la version du serveur)

Commentaires mis en en-tête (séparer les lignes par «\n») :

Inclut un horodatage de création, mise à jour et dernière vérification des bases de données

Utiliser le mode transactionnel

Désactiver la vérification des clés étrangères

Exporter les vues comme des tables

Maximiser la compatibilité avec un système de base de données ou un ancien serveur MySQL :

structure

données

structure et données

Pour les options de créations d'objets cochez tous les énoncés :

Options de création d'objets

Ajouter les énoncés :

- Ajouter un énoncé CREATE DATABASE / USE
- Ajouter un énoncé DROP TABLE / VIEW / PROCEDURE / FUNCTION / EVENT / TRIGGER
- Ajouter un énoncé CREATE TABLE
- Ajouter un énoncé CREATE VIEW
- Ajouter un énoncé CREATE PROCEDURE / FUNCTION / EVENT
- Ajouter un énoncé CREATE TRIGGER
- Options pour CREATE TABLE :
 - IF NOT EXISTS
 - AUTO_INCREMENT

Entourer les noms des tables et des colonnes par des guillemets obliques (*Protège les noms de tables et de colonnes comportant des caractères spéciaux ou des mots réservés*)

Ensuite cocher les deux modes pour les options de création de données :

Options de création de données

Vider la table avant d'insérer

Au lieu d'énoncés INSERT, utiliser :

- Énoncés INSERT DELAYED
- Énoncés INSERT IGNORE

Fonction à utiliser lors de l'exportation des données :

Syntaxe à utiliser lors de l'insertion de données :

- inclure les noms de colonnes dans chaque énoncé INSERT
Par exemple: `INSERT INTO tbl_name (col_A,col_B,col_C) VALUES (1,2,3)`
- insérer des lignes multiples avec chaque énoncé INSERT
Par exemple : `INSERT INTO tbl_name VALUES (1,2,3), (4,5,6), (7,8,9)`
- les deux modes ci-dessus
Par exemple : `INSERT INTO tbl_name (col_A,col_B, col_C) VALUES (1,2,3), (4,5,6), (7,8,9)`
- aucun des modes ci-dessus
Par exemple : `INSERT INTO tbl_name VALUES (1,2,3)`

pour ma part j'ai laissé la taille par défaut de la requête générale :

Taille maximum de la requête générée

Exporter les colonnes binaires en format hexadécimal (par exemple, «abc» devient 0x616263)

Exporter les colonnes TIMESTAMP en UTC (permet les colonnes TIMESTAMP d'être exportées et importées entre des serveurs de zones horaires différentes)

Puis enfin valider votre action et enregistrer la base dans un dossier de votre choix.

Notre base a été exportée avec succès, au tour maintenant de la migration de celle ci.

Par défaut dans phpMyAdmin l'upload de documents est limité à 4 méga octets, pour ce faire nous allons rentrer dans notre serveur et augmenter la limitation dans le php.ini, après avoir ouvert une fenêtre de console avec les droits admin tapez les lignes suivantes :

```
localhost:/home/jro # vi /etc/php5/apache2/php.ini
```

Une fois dans le php.ini, il suffira de trouver la ligne qui nous incombe et d'en augmenter la taille en insérant la nouvelle donnée en appuyant au préalable sur « i » pour insérer, **prenez garde de ne pas vous tromper si vous n'avez pas fait une copie de votre php.ini**

Dans la ligne upload_max_filesize augmentez la limitation, dans mon exemple à 256M :

```
;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;

; Whether to allow HTTP file uploads.
; http://php.net/file-uploads
file_uploads = On

; Temporary directory for HTTP uploaded files (will use system default if not
; specified).
; http://php.net/upload-tmp-dir
upload_tmp_dir =

; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize = 256M

; Maximum number of files that can be uploaded via a single request
max_file_uploads = 20
```

Enregistrez ensuite la modification en sortant de l'insertion en appuyant sur ' : ', puis 'w' pour écrire la modification et enfin 'q' pour quitter l'éditeur vi.

Vous pouvez désormais importer votre base de donnée.

Pour finir nous allons adapter l'ancienne base de donnée au nouveau site via une simple requête SQL comme suit :



Les requêtes en clair :

```
UPDATE wp_options  
SET option_value = replace(option_value, 'http://www.ancien-site.com',  
'http://www.nouveau-site.com')  
WHERE option_name = 'home'
```

```
UPDATE wp_posts  
SET guid = REPLACE (guid, 'http://www.ancien-site.fr', 'http://www.nouveau-site.fr');
```

```
UPDATE wp_posts  
SET post_content = REPLACE (post_content, 'http://www.ancien-site.fr',  
'http://www.nouveau-site.fr');
```

Veillez à respecter la syntaxe, mysql est pointilleux.

Executez enfin les requêtes.

Nous voilà avec une base de données à jour pour notre nouveau site wordpress.

Lors de votre prochain login à votre site, vous aurez une éventuelle mise à niveau de votre base de donnée.

[Source](#)