SCI6306 Bases de données documentaires (Automne 2023)

Christine Dufour, EBSI, UdeM

A2023 27 octobre 2023

Cours 7 : Création d'une base de données dans phpMyAdmin

SCI6306

Christine Dufour, EBSI, UdeM

Table des matières

I Création d'une base de données avec phpMyAdmin

- + Au programme aujourd'hui
- Création d'une base de données dans phpMyAdmin
    - Création d'une table de données
    - Définition des propriétés des champs
    - Définition des index
    - Définition des relations entre les tables
    - Génération du dictionnaire de données
- Saisie des données dans phpMyAdmin
    - Saisie directe des données dans phpMyAdmin
    - Importation en lot d'enregistrements
- Exportation de données dans phpMyAdmin
- Projet de session, volet B
- Ressources en lien avec le cours

1. + Au programme aujourd'hui

- **Création d'une BD avec phpMyAdmin**
    - Création d'une table de données
    - Définition des propriétés des champs
    - Définition des index
    - Définition des relations entre les tables
- **Saisie de données**
- **Exportation des données / de la base de données**
- **Projet de session**
    - Remise du volet A (au début du cours)
    - Présentation du volet B

2. Création d'une base de données dans phpMyAdmin

La création d'une base de données dans phpMyAdmin implique différentes étapes décrites ci-dessous : il faut y définir, dans cet ordre, les tables, leurs champs, les index associés et finalement les relations entre les tables.

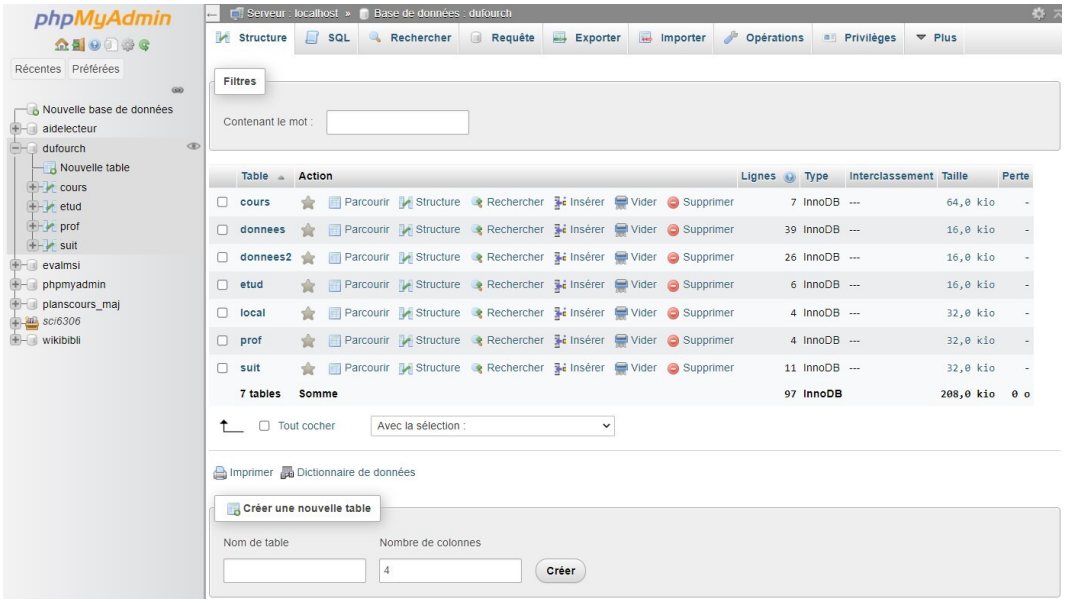
Remarque **:**

Le Guide abrégé d'utilisation de phpMyAdmin 5.2.0[1] est un complément à la matière ici présentée.

[1] https://cours.ebsi.umontreal.ca/sci6306/co/site_phpmyadmin.html

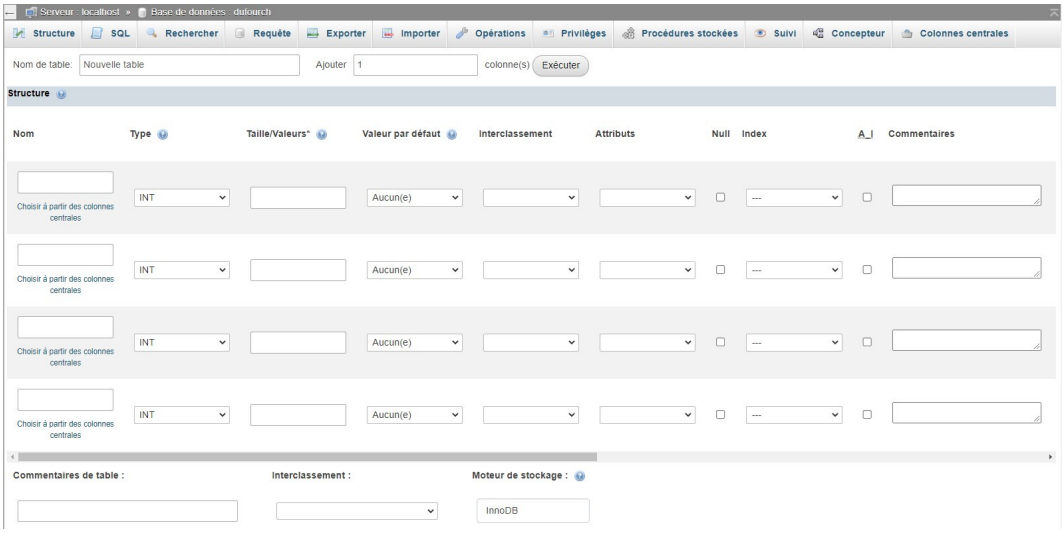
2.1. Création d'une table de données

La création d'une table de données se fait dans phpMyAdmin en cliquant premièrement sur la base de données où l'on veut ajouter la table et, dans l'onglet **Structure**, en indiquant le nom de la table à ajouter ainsi que le nombre de champs (colonnes) de cette dernière dans la zone **Créer une nouvelle table** au bas de l'écran.



Onglet Structure pour une base de données dans phpMyAdmin

Lorsque l'on clique par la suite sur **Créer**, on se retrouve à l'interface nous permettant de définir les caractéristiques de la table ainsi que de ses champs.



Ajout d'une table dans phpMyAdmin

Les principaux éléments génériques à définir sont les suivants :

- **Nom** de la table : nom assez court et représentatif du contenu de la table
- **Commentaires** sur la table : description succincte du contenu de la table (complément au nom)
- **Interclassement** : jeu de caractères et caractéristiques du classement
    - Par exemple, *utf8_general_ci* = Unicode (multilingue), insensible à la casse
    - *Note* : à définir si on ne veut pas retenir l'interclassement défini au niveau de la base de données (ce qui est relativement rare)

- **Moteur de stockage**
    - Selon les besoins sur le plan de la performance et de la protection de l'intégrité des données (entre autres)
    - Le plus fréquemment utilisé : *InnoDB* (celui utilisé par défaut lorsque l'on crée une table)

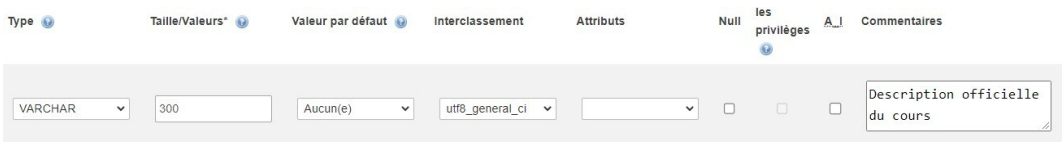
2.2. Définition des propriétés des champs

Après avoir défini les caractéristiques génériques de la table, il faut définir les propriétés de ses différents champs en remplissant les paramètres pertinents. Les principales propriétés à définir sont les suivantes :

- **Nom** du champ (*conseil* : rester simple!)
- **Type de données** (voir dans la documentation complémentaire la section Principaux types de champ sous MySQL[1])
- **Taille** du champ ou **valeurs prédéfinies** (selon le type de données)
- **Valeur par défaut** (s'il y a lieu)
- **Interclassement** (s'il diffère de celui de la table)
- **Attributs particuliers** (en particulier « *unsigned* » pour les champs numériques qui n'acceptent que les valeurs positives)
- Acceptation ou non de la **valeur nulle** (acceptation = champ facultatif)
- Champ dont la **valeur s'incrémente automatiquement** (A_I)
- **Commentaires** (pour rappeler, par exemple, les principales règles d'écriture)

***Exemple* : Propriétés du champ DESCR dans la table COURS de la base de données INSCRIP**

Le champ *DESCR* de la table COURS de la base de données *INSCRIP*, dans le schéma relationnel de la BD, a pour caractéristiques d'être de type **Caractère**, d'une taille de **300** et d'être **obligatoire**. Voici comment ces caractéristiques se traduisent dans phpMyAdmin :



Propriétés du champ DESCR de la table COURS de la base de données INSCR

À noter :

- La longueur d'une description étant variable, le type VARCHAR a été privilégié pour éviter d'enregistrer inutilement des espaces vides afin d'arriver à la taille maximale de 300.
- La taille a été définie à 300.
- Comme il n'y a pas de "valeur type" possible pour une description, aucune valeur par défaut n'est précisée. Un exemple où une valeur par défaut pourrait être pertinente est le cas d'un champ PAYS dans une base de données où l'on sait que la majorité des enregistrements auront pour valeur CANADA. Afin d'accélérer la saisie, on peut définir CANADA comme valeur par défaut qui sera ainsi inscrite automatiquement lors de la création d'un nouvel enregistrement et modifiable, bien entendu, au besoin par la suite.
- Comme les données du champ DESCR seront dans la même langue que l'ensemble de la base de données, l'interclassement par défaut - c'est-à-dire celui de la table de données qui est en fait celui de la base de données au complet - a été conservé.
- Aucun attribut particulier n'est nécessaire.
- La case pour *Null* n'a pas été cochée. Le champ étant obligatoire, on ne veut en effet pas permettre la saisie de la valeur NULL.

[1] https://cours.ebsi.umontreal.ca/sci6306/co/MySQL_types_champs.html

- Le contenu de ce champ ne consistant pas à un chiffre auto-incrémenté, la colonne A_I n'a pas été cochée.
- Finalement, une courte description de la nature du champ a été ajoutée dans les commentaires associés à ce dernier.

2.3. Définition des index

En sus de définir, pour une table, les propriétés de ses champs, il faut définir le ou les index qui seront **utiles** et **nécessaires** au bon fonctionnement de la base de données. Les index servent à différents plans :

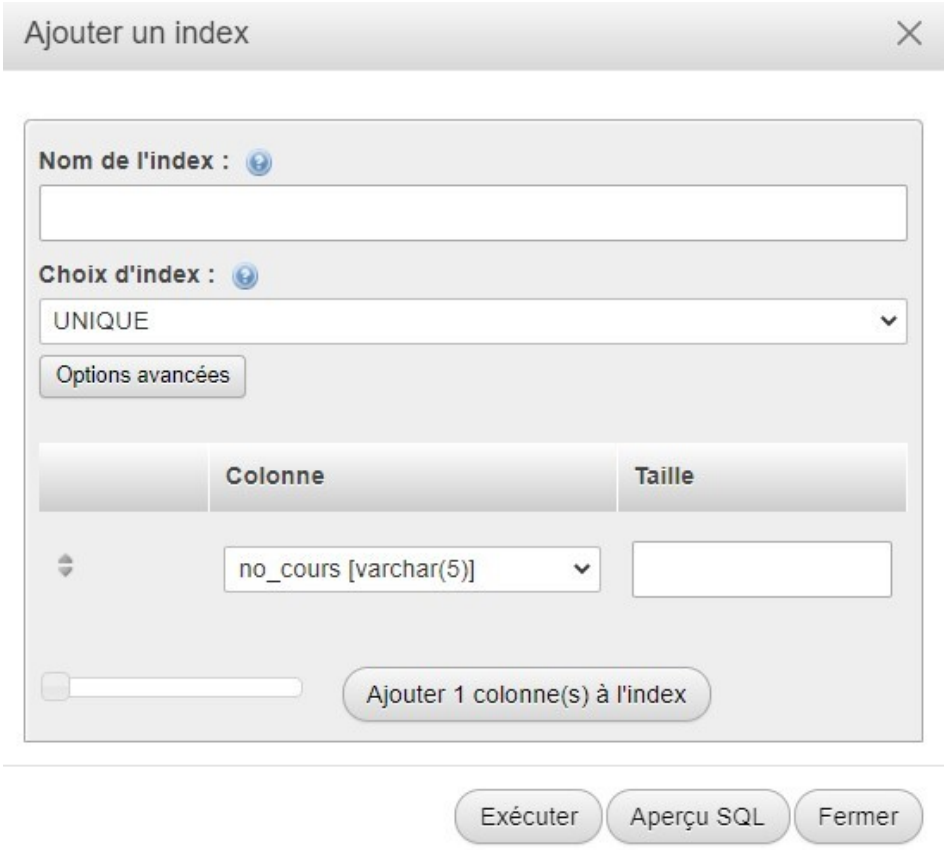
- Bien que l'indexation d'un champ ne soit pas nécessaire pour qu'il soit cherchable (sauf pour la recherche plein texte), elle permet d'**accélérer la recherche**. Il faut donc prendre soin d'indexer les champs où l'on prévoit des *recherches fréquentes.*
- Les index sont **obligatoires** dans certains contextes :
    - Pour les *clés primaires* (pour contrôler l'unicité de leurs valeurs dans une table),
    - Pour les *champs servant de clés externes* (pour s'assurer qu'un enregistrement dans une table liée a bien une correspondance dans la table principale),
    - Pour les *champs à valeurs uniques* (pour contrôler l'unicité de leurs valeurs),
    - Pour les champs où l'on souhaite faire de la *recherche plein texte*.

On retrouve quatre principaux **types** d'index :

- *Primary* : pour une clé primaire,
- *Unique* : pour avoir des valeurs uniques dans un champ autre que la clé primaire,
- *Fulltext* : pour la recherche plein texte,
- *Index* : index-mots sans caractéristique particulière (entre autres pour les clés externes).

Un index peut se faire **sur un seul champ** comme il peut en **regrouper plusieurs**. C'est le cas, par exemple, d'une clé primaire multichamp qui regroupera dans un index tous les champs qui la composent.

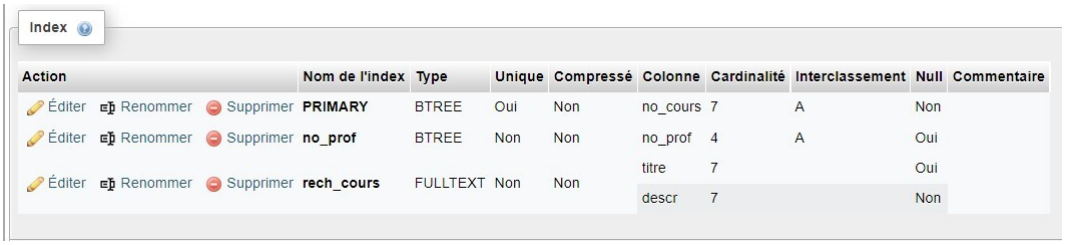
L'interface pour la définition des index est accessible à partir de l'onglet **Structure** d'une table, sous la table présentant ses différents champs. Lorsque l'on clique sur **Exécuter** à droite de "Créer un index sur...", l'interface proposée nous permet de nommer l'index à ajouter, d'en choisir le type, de préciser le ou les champs (colonnes) qui le constituent :



Interface pour la définition des index

Exemple : Index définis pour la table COURS de la base de données INSCRIP

La table COURS de la base de données INSCRIP possède trois index :



Index définis pour la table COURS de la base de données INSCRIP

À noter :

- Le premier index nommé **PRIMARY** est celui de la clé primaire de la table. Comme cette clé primaire est composée d'un seul champ (NO_COURS), il n'y a qu'un seul champ précisé pour cet index. Puisqu'il s'agit d'une clé primaire, l'unicité des valeurs est contrôlée. Dans cet affichage, seul le type générique d'index est indiqué, soit ici BTREE. Pour voir le type spécifique, il suffit de cliquer sur *Éditer*. Cet index est de type *Primary*. Il est créé automatiquement lorsque l'on définit la clé primaire d'une table. La création d'une clé primaire dans une table se fait dans l'onglet **STRUCTURE**, en cochant le ou les champs qui serviront de clé primaire et en cliquant sur **Primaire** qui se trouve sous la table des champs.
- Le deuxième index nommé **no_prof** correspond à un index pour une clé externe. Le champ NO_PROF sert en effet pour relier un cours à la table PROF afin d'obtenir les informations sur le professeur ou la professeure associée à ce cours. Comme un.e professeur.e peut enseigner plusieurs cours, l'unicité des valeurs n'est pas recherchée. Cet index est de type *Index*. Il faut créer les index sur les champs de clé

externe avant de définir les relations entre les tables. La création de ce type d'index peut se faire dans l'onglet **STRUCTURE**, en cochant le champ à indexer et en cliquant sur *Index* qui se trouve sous la table des champs.

- Le troisième index nommé **rech_cours** a été défini pour permettre les recherches plein texte sur les champs à contenu textuel plus dense pour lesquels on pressentait l'utilité de ce type de recherche. Pressentant des recherches par thématique, il a été décidé de joindre les deux champs TITRE et DESCR dans un seul index de type *Fulltext*. Cet index peut aussi se créer à partir de l'onglet **STRUCTURE**, en cochant le ou les champs à indexer et en cliquant sur *Texte entier* qui se trouve sous la table des champs.
- Comme des recherches fréquentes ne sont pressenties que pour les champs TITRE et DESCR, qui sont déjà indexés, aucun autre index n'a été ajouté à cette table.

2.4. Définition des relations entre les tables

Maintenant que l'ensemble des éléments constitutifs des tables est défini, il est possible de procéder à la définition des relations entre les tables. Bien qu'une base de données puisse en théorie être fonctionnelle sans définir les relations explicitement, dans la pratique, l'absence de relations provoquera des **problèmes sérieux** sur le plan de la **cohérence** des données. Ce sont les relations qui permettent de maintenir cette cohérence en permettant la mise à jour des données liées entre deux tables. Par exemple, dans INSCRIP, c'est la présence d'une relation entre les tables ETUD et SUIT qui permet d'automatiquement effacer les inscriptions à un cours si l'on supprime un.e étudiant.e de la table ETUD, ou de modifier le numéro d'étudiant.e dans la table SUIT suite à une modification du numéro dans la table ETUD. Différents types "d'interaction" sont possibles entre deux tables liées. Il y a deux contextes où il faut préciser le type d'interaction pour une relation entre deux tables : (1) ce qu'il faut faire si on **met à jour** le(s) champ(s) liant les deux tables (*on update*), et (2) ce qu'il faut faire si on **efface** un enregistrement dans une table liée (*on delete*).

Trois **types d'interaction** sont possibles :

- **CASCADE** : les actions sur la table principale se répercutent sur la table liée
    - Par exemple, dans INSCRIP, si on supprime un cours, toutes ses inscriptions sont effacées dans SUIT.
    - Permet de préserver l'intégrité référentielle des données en s'assurant que chacune des valeurs de la clé externe de la table liée correspond à un enregistrement de la table principale.
- **SET NULL** : si on supprime un enregistrement de la table principale, les enregistrements correspondant dans la table liée ne sont pas effacés, mais la valeur de la clé externe est changée pour NULL
    - Par exemple, dans INSCRIP, on peut se demander si c'est une bonne idée lorsque l'on efface un.e professeur.e de voir tous ses cours être effacés en cascade... Peut-être serait-il préférable de plutôt mettre la valeur NULL dans le champ NO_PROF de la table COURS. Une condition nécessaire pour cette action est bien entendu que le champ accepte les valeurs NULL (donc soit facultatif), ce qui est bien le cas pour NO_PROF dans la table COURS.
- **RESTRICT** : on ne peut pas modifier la clé primaire ou effacer un enregistrement de la table principale s'il existe un enregistrement correspondant dans la table liée

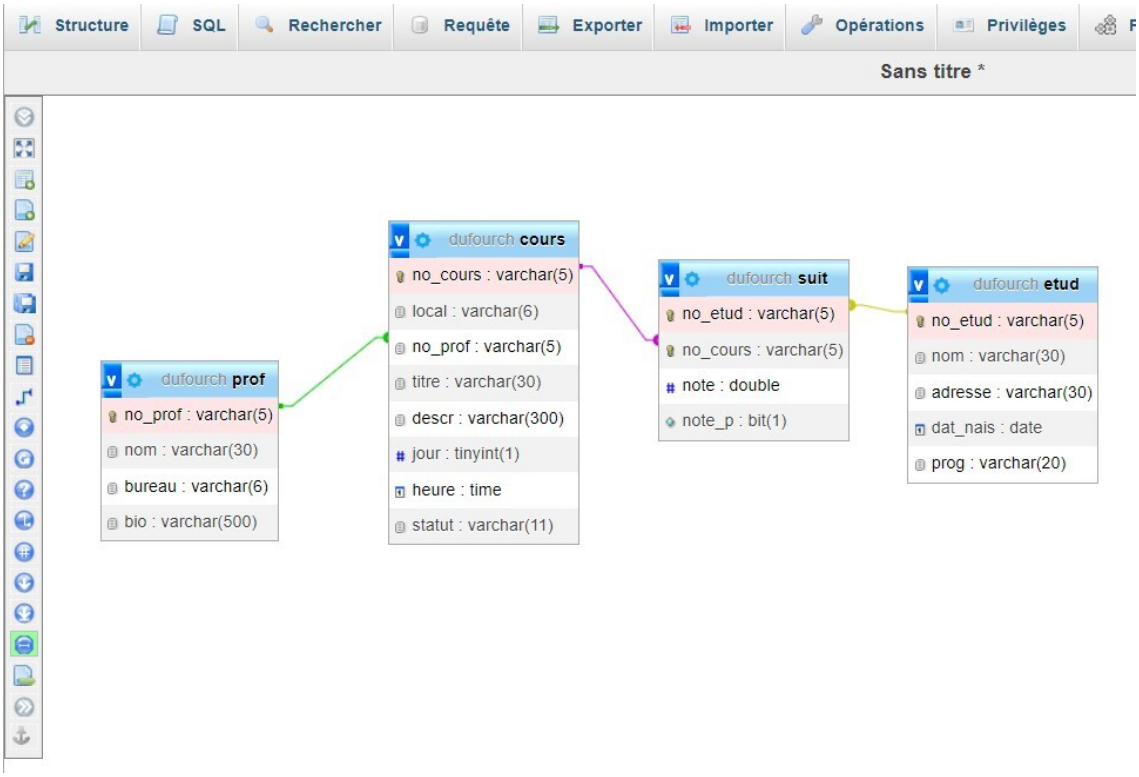
La définition des relations peut se faire soit en mode plus graphique en utilisant le **Concepteur**, ou en mode plutôt textuel à partir de l'onglet **Structure** d'une table en cliquant sur **Vue relationnelle**. Un des avantages d'utiliser la vue relationnelle de la table où se trouve la clé externe est qu'il est possible d'y définir des contraintes de clé externe portant sur plusieurs champs simultanément (ce qui est le cas lorsque la clé primaire de la table liée est une clé multichamp). De plus, la vue relationnelle indique explicitement le type d'interaction définie pour la mise à jour et la suppression d'enregistrement.

⚠ *Attention* :

Il est bien important d'attendre d'avoir bien tout défini au niveau des tables, en particulier de s'assurer que les champs servant de clés externes ont bien été définis exactement du même type de données dans les deux tables liées, ainsi que les index bien définis pour ces clés externes. Procéder un peu trop par essai-erreur lors de la définition des relations dans une base de données peut entraîner une certaine **instabilité** de la base de données, voire même la **corrompre**.

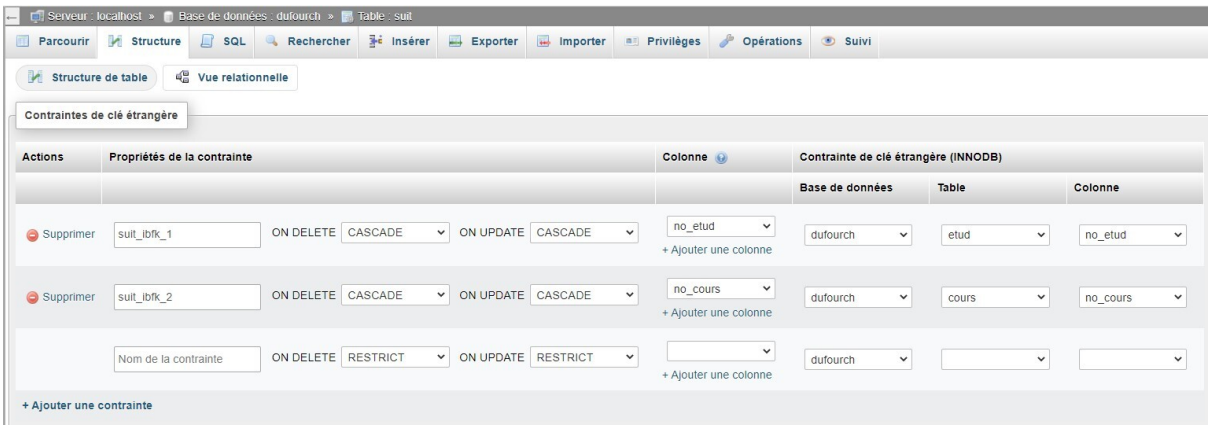
🔍 *Exemple* : Relations définies pour la table SUIT de la base de données INSCRIP

La table SUIT de la base de données INSCRIP est liée à deux tables, soit la table COURS et la table ETUD :



Visualisation des relations de la base de données INSCRIP avec le Concepteur

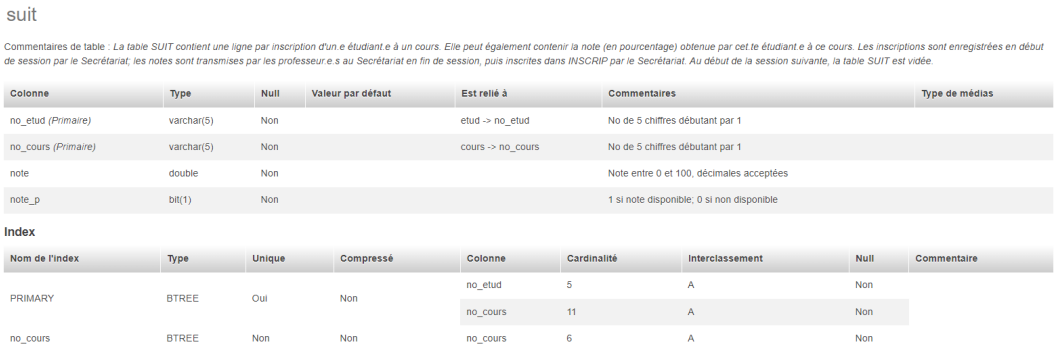
Si on regarde plus précisément les types de relation définis, on remarque que tant pour la modification que pour la suppression, l'action retenue est de répercuter (cascade) sur SUIT les actions faites sur les tables COURS et ETUD. Ainsi, si on supprime l'enregistrement pour un.e étudiant.e, toutes les inscriptions de cet.te étudiant.e disparaissent de la table SUIT. De même, si on change le numéro d'un cours, ce dernier sera modifié dans la table SUIT.



Visualisation des relations pour la table SUIT de la base de données INSCRIP (Vue relationnelle)

2.5. Génération du dictionnaire de données

Il est possible de générer, à partir de l'onglet **Structure** d'une base de données, un "**dictionnaire de données**" où on retrouve, pour chaque table, le détail sur ses champs et ses index. Cette synthèse des caractéristiques des tables, de leurs champs et de leurs index peut se révéler fort utile lorsque vient le temps de vérifier que la structure définie correspond bien au schéma relationnel de la BD.



suit

Commentaires de table : *La table SUIT contient une ligne par inscription d'un·e étudiant·e à un cours. Elle peut également contenir la note (en pourcentage) obtenue par cet·te étudiant·e à ce cours. Les inscriptions sont enregistrées en début de session par le Secrétariat; les notes sont transmises par les professeur·e·s au Secrétariat en fin de session, puis inscrites dans INSCRIP par le Secrétariat. Au début de la session suivante, la table SUIT est vidée.*

Colonne	Type	Null	Valeur par défaut	Est relié à	Commentaires	Type de médias
no_etud *(Primaire)*	varchar(5)	Non		etud -> no_etud	No de 5 chiffres débutant par 1	
no_cours *(Primaire)*	varchar(5)	Non		cours -> no_cours	No de 5 chiffres débutant par 1	
note	double	Non			Note entre 0 et 100, décimales acceptées	
note_p	bit(1)	Non			1 si note disponible; 0 si non disponible	

Index

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	no_etud	5	A	Non	
				no_cours	11	A	Non	
no_cours	BTREE	Non	Non	no_cours	6	A	Non	

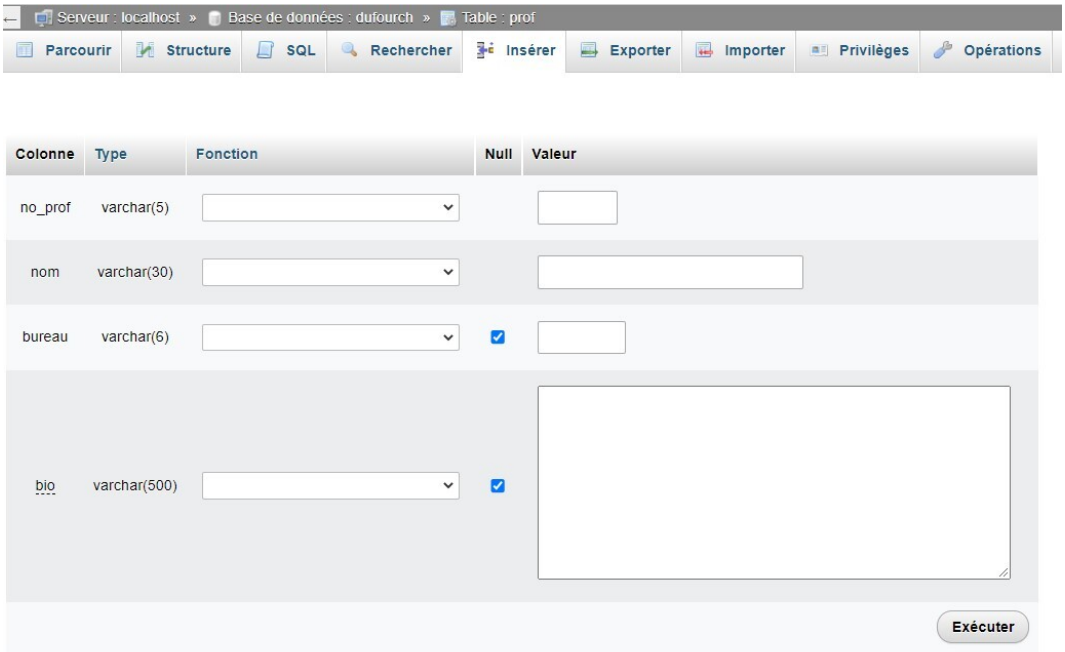
Extrait du dictionnaire de données de la base de données INSCRIP

3. Saisie des données dans phpMyAdmin

La saisie des données dans phpMyAdmin peut se faire de deux manières, soit par une **saisie directe des données**, soit par une **importation en lot de données**.

3.1. Saisie directe des données dans phpMyAdmin

La saisie directe des données permet d'**ajouter** de nouveaux enregistrements ainsi que de **modifier**, voire même de **supprimer**, des enregistrements existants. Pour ajouter un nouvel enregistrement, il s'agit de sélectionner la table et de cliquer sur l'onglet **Insérer**. Un formulaire de saisie s'affichera offrant la possibilité d'ajouter deux enregistrements. Les contrôles pour la saisie retrouvés dans le formulaire seront choisis en fonction du type de données des champs (longueur des boîtes de saisie par exemple).



Ajout d'enregistrements dans une table dans phpMyAdmin

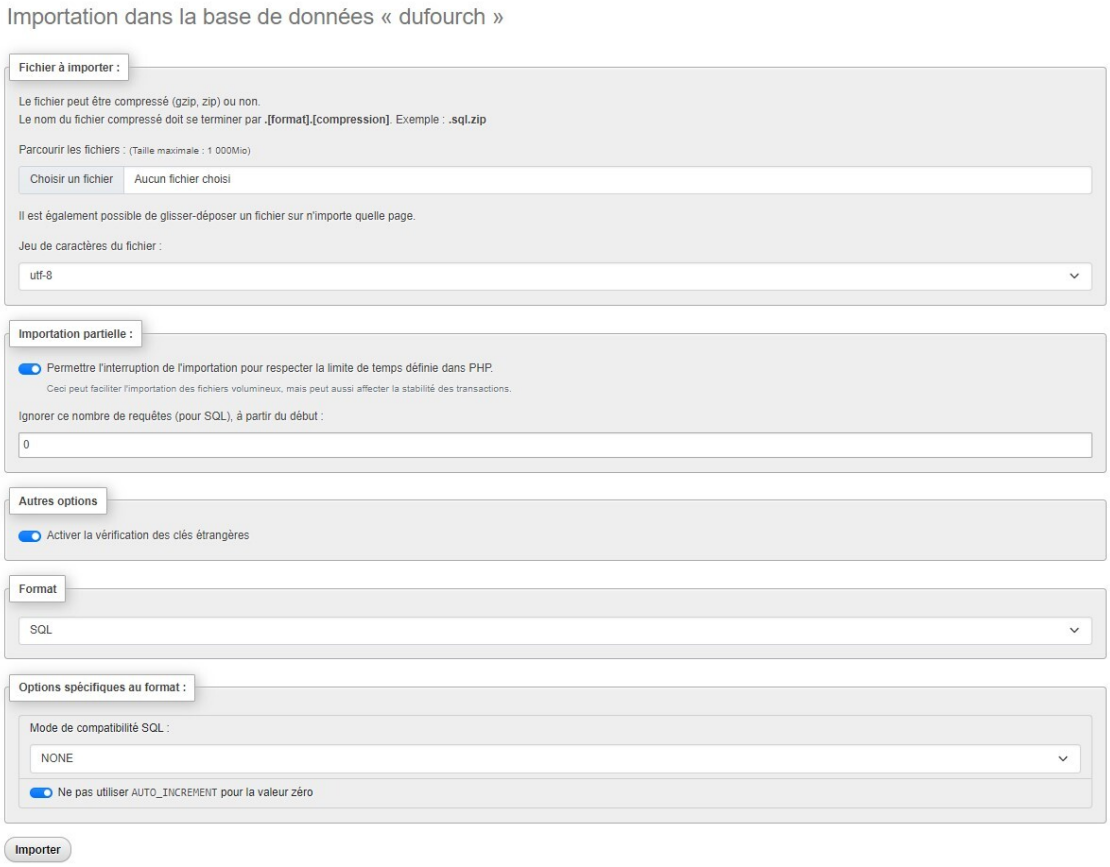
Pour **modifier** les informations dans un enregistrement, on peut soit directement cliquer sur la valeur à modifier dans la table, soit cliquer sur **Éditer** à gauche de l'enregistrement à modifier pour avoir accès au formulaire de saisie. Pour supprimer un enregistrement, il suffit de cliquer sur **Supprimer** à gauche de l'enregistrement désiré. Il est à noter que la suppression est définitive.

3.2. Importation en lot d'enregistrements

Dans certains contextes, par exemple une migration d'un autre environnement vers une base de données MySQL, il est possible de procéder à l'importation d'un seul coup des données existantes; c'est ce qu'on appelle de l'**importation en lot**. Les données peuvent être importées dans une **table qui existe déjà** dans la base de données comme il est possible de le faire dans une **nouvelle table**.

Plusieurs formats sont supportés pour l'importation dont SQL, CSV, XML et ODS (*opendocument spreadsheet*). Par exemple, pour l'importation d'un classeur ODS, il faut s'assurer que :

- La **feuille** où se trouvent les données ait le **même nom** que le nom de la **table** (pour importer dans une table existante).
- Que les **noms des champs** se retrouvent dans la **première ligne** de la feuille de données. Les enregistrements se retrouveront dans les lignes suivantes.
- Que, si on procède à une **importation dans une table existante**, les données correspondent aux **caractéristiques des champs définis** (par exemple, le type de données, le caractère obligatoire/facultatif, la longueur des champs). Si les caractéristiques ne sont pas compatibles, il y aura un message d'erreur lors de l'importation.



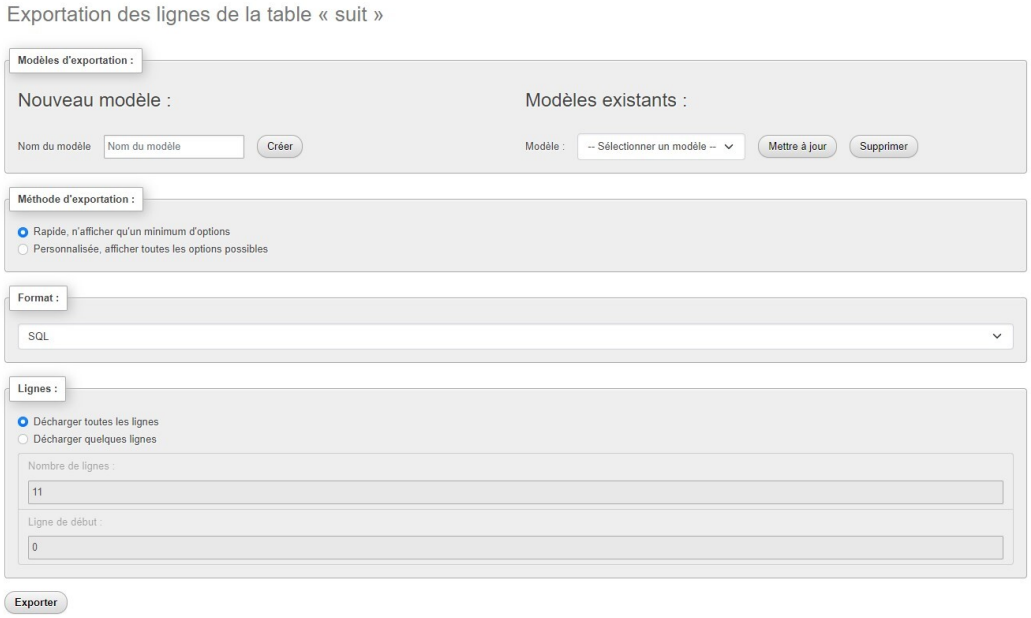
Importation de données dans une table ou une base de données dans phpMyAdmin

Dans le scénario d'une importation en lot d'enregistrements dans une structure existante, l'étape de la préparation des données en est une d'importance. Cette préparation peut se faire manuellement, dans certains cas, bien que cela ne soit pas toujours la manière la plus efficace de procéder. On peut exploiter les différentes fonctions d'un tableur comme *Excel* pour procéder à certaines transformations. On peut aussi utiliser des outils spécialisés dans ces transformations comme **OpenRefine**. Un mode d'emploi abrégé[1] pour ce dernier est accessible sur le site du cours.

[1] https://cours.ebsi.umontreal.ca/sci6306/co/site_guide_openrefine.html

4. Exportation de données dans phpMyAdmin

Il est possible d'exporter les données d'**une table** d'une base de données dans différents formats (CSV, PDF, SQL, ...). Il suffit de cliquer sur la table, ensuite sur l'onglet **Exporter** et de finalement choisir les paramètres d'exportation.



Exportation de table de données dans phpMyAdmin

Il est aussi possible d'exporter **toute la base de données**, entre autres en format SQL, pour pouvoir la réimporter dans phpMyAdmin. Cela peut se révéler utile pour prendre des copies de sauvegarde d'une base de données pendant sa conception. Cette exportation est possible en cliquant la base de données à exporter et en accédant à l'onglet **Exporter** afin de définir les paramètres d'exportation.

5. Projet de session, volet B

L'objectif du deuxième volet du projet de session est **développer la base de données** correspondant au schéma relationnel fourni ainsi que de **saisir des informations** dans les tables de la base de données créée.

Deux **livrables** seront à remettre :

- La **base de données MySQL** correspondant au schéma fourni et contenant les données demandés
- Un **journal de bord** où sont consignés les différents éléments demandées (justification des choix, réflexion sur la qualité de la base de données, etc.)

Le protocole du volet B sera déposé sur StudiUM à la pause du cours (si toutes les équipes ont remis le volet A). Pour ce travail, vous utiliserez le **compte d'équipe sur phpMyAdmin** pour développer la base de données. Vous pouvez aussi utiliser l'espace sur GIN-EBSI de l'équipe pour conserver une copie de sauvegarde du journal de bord.

6. Ressources en lien avec le cours

Matériel de cours

- *Notes de cours [cf. sci6306_cours7_notes.pdf]*