TP 1 : Création de la base de données

Dans un premier temps, nous allons créer la base de travail. Il s'agit des relations concernant les notes des étudiants.

1) En utilisant l'onglet Bases de données, créer la base base_etudiants.

2) Dans cette base, toujours à l'aide de l'interface (onglet Structure), créer les tables

- ✓ etudiants (clé primaire num_etu)
- ✓ notes (clé primaire composée sur les champs _num_etu et _num_mat)
- ✓ matieres (clé primaire num_mat)
- ✓ enseignants (clé primaire num_ens ; index sur le champ nom_ens)

On veillera à choisir soigneusement le type de chaque champ, en sélectionnant le plus approprié parmi les types disponibles sous MySQL.

Exercice 1 :

Indiquer, sous forme de tableau et en les justifiant, les propriétés choisies pour chaque champ : type, taille/valeur, valeur par défaut, attributs, NULL autorisé ?, index, AUTO_INCREMENT.

TP 2 : Insertion de données

Le but ici est de se familiariser avec l'interface du logiciel et d'étudier quelques propriétés des champs.

1) En utilisant l'onglet Insérer, saisir manuellement l'étudiant n°2 (dans la table appropriée). Noter la présence d'un espace entre le nom et le prénom des étudiants.

Exercice 1 :

Recopier et expliquer la requête générée par MySQL lors de cette insertion.

2) Vérifier que l'incrémentation automatique est bien fonctionnelle dans la table etudiants (propriété AUTO_INCREMENT (A_I) pour la clé num_etu), en ajoutant l'étudiante Houcine, Fatima, mais cette fois sans spécifier son numéro.

Exercice 2 :

Recopier et expliquer la requête générée par MySQL lors de cette insertion, ainsi que son résultat.

3) Interdire les doublons dans le champ nom_etu (en lui ajoutant un index UNIQUE dans l'onglet Structure) et vérifier que cette contrainte est respectée.

Exercice 3 :

Recopier et expliquer le message délivré par MySQL lors de l'insertion d'un doublon pour le nom d'étudiant.

TP 3 : Importation et exportation de données

On montre ici comment importer des données à l'aide de l'interface ou d'une requête, et comment exporter une base complète à destination d'un autre SGBD ou d'un tableur.

1) Activer la table etudiants puis, dans l'onglet Importer, importer le fichier etudiant.csv. Ce fichier est au format CSV, le séparateur de colonnes est le caractère tabulation (noté \t), et les colonnes ne sont entourées par aucun caractère particulier. Cette importation devrait générer une erreur !

Exercice 1 :

Expliquer l'origine de cette erreur, le résultat obtenu quant au contenu de la table etudiants, et proposer des solutions possibles pour y remédier (réaliser l'une d'entre elles).

2) Une autre façon d'importer des données dans une table existante est d'utiliser la clause SQL :

LOAD DATA INFILE 'fichier' INTO TABLE table

Dans l'onglet SQL, réaliser de cette façon l'importation des 3 autres fichiers .csv correspondant aux tables autres qu'etudiants. Pour cela, vous avez intérêt à copier ces fichiers dans le répertoire contenant la base (soit C:\wamp\bin\mysql\mysql5.5.24\data\base_etudiants\), ce qui évite de spécifier le chemin du fichier dans la requête LOAD DATA.

Exercice 2 :

En vous référant à la documentation MySQL sur LOAD DATA, faire le parallèle entre l'importation de la table etudiants par cette méthode et celle utilisée à la question précédente.

3) Dans l'onglet approprié, exporter le code SQL de la base complète (structure et données).

Exercice 3 :

Prendre connaissance des commandes SQL contenues dans le fichier ainsi exporté. Commenter ce code, tout particulièrement la structure des requêtes CREATE TABLE.

4) On cherche cette fois à exporter la base pour Excel, d'une part, et pour le tableur Calc de LibreOffice (Open Document Spreadsheet), d'autre part.

Exercice 4 :

Quel format préconisez-vous dans chaque cas ? Justifiez ce choix lorsque plusieurs solutions sont possibles, comme dans le cas de l'exportation pour Excel.

TP 4 : Sélection de données

Nous allons ici pratiquer la sélection de données mono-table et multi-tables grâce au langage SQL.

Exercice 1 :

Pour chaque question,

• reproduire le code de la (des) requête(s), en expliquant les principales difficultés, en particulier celles qui sont nouvelles par rapport aux requêtes précédentes ;

• donner des indications permettant de vérifier si le résultat est correct, par exemple : nombre de lignes, valeur(s) retournée(s), etc.

1) Trouver les noms et dates de naissance des étudiantes nées avant 1994. Utiliser les 2 formats de date possibles sous MySQL.

2) Trouver la note de l'étudiante Houcine, Fatima en Bases de Données. Les champs texte, tels que le nom de l'étudiant ou celui de la matière, sont-ils sensibles à la casse et aux accents ?

3) Trouver le nombre d'étudiants nés en 1994. On pourra pour cela utiliser la fonction YEAR().

4) Trouver (s'il y en a) les enseignants qui n'enseignent aucune matière. Pour cela, on pourra formuler successivement les requêtes suivantes :

• retrouver les enseignants ainsi que la (les) matière(s) que chacun enseigne ;

• transformer la jointure de la requête précédente en jointure externe, afin de retrouver tous les enseignants, qu'ils enseignent ou non une matière ;

• ne retenir finalement que les enseignants qui n'enseignent aucune matière.

5) Trouver les numéros, nom et moyenne des étudiants de moins de 22 ans (ou dont la date de naissance est inconnue) ayant eu une note dans chacune des matières, en affichant les noms par ordre croissant.

TP 5 : Mise à jour de données

Ici sont étudiées de nouvelles clauses SQL, qui permettent la modification des données de la base.

1) Afficher la table des enseignants et, grâce au bouton, mettre à 0 l'ancienneté de Taha, Moussa.

Exercice 1 :

Recopier la requête générée par MySQL lors de cette mise à jour, et en déduire la syntaxe de cette nouvelle clause.

2) En vous inspirant de la requête précédente, formuler une requête pour mettre à 0 l'ancienneté de tous les enseignants ayant le grade MCB.

Exercice 2 :

Recopier cette requête. Quel est le nombre de lignes ainsi modifiées ?

3) Formuler une requête mise à jour multi-tables (voir au besoin le manuel MySQL) pour ajouter un point à toutes les notes obtenues en Compilation.

Exercice 3 :

Recopier cette requête et expliquer sa syntaxe. Quel est le nombre de lignes affectées ?

TP 6 : Suppression de données

Hormis de nouvelles clauses SQL permettant la suppression de données dans la base, nous abordons ici le problème de la cohérence de ces données.

1) Supprimer le dernier étudiant grâce au bouton.

Exercice 1 :

Recopier la requête générée par MySQL lors de cette suppression. En déduire la syntaxe de cette nouvelle clause.

2) En vous inspirant de la requête précédente, supprimer toutes les étudiantes nées avant 1990.

Exercice 2 :

Recopier cette requête. Quel problème pose-t-elle quant à la cohérence des données de la base ?

3) Formuler une requête multi-tables permettant de supprimer toutes les informations relatives à l'étudiant Dupont, Charles.

Exercice 3 :

Recopier et expliquer cette requête. Quel avantage présente-t-elle (par rapport à la précédente) quant à la cohérence des données de la base ?

4) Formuler une requête multi-tables permettant de supprimer les notes de l'étudiant Dubois, Jules sans supprimer cet étudiant lui-même.

Exercice 4 :

Recopier cette requête et expliquer sa syntaxe en la comparant à la précédente. La cohérence des données relatives à cet étudiant est-elle préservée ?

TP 7 : Relations entre tables

La gestion des relations est un mécanisme puissant permettant de gérer l'intégrité référentielle de la base. Il consiste à contraindre une clé étrangère (« foreign key », en anglais) à prendre l'une des valeurs de la clé primaire (« primary key ») qu'elle référence. Ce mécanisme est pris en charge par le moteur de stockage de table InnoDB (moteur par défaut sous MySQL depuis la version 5.5.5), moyennant quelques prérequis sur les propriétés des champs clés étrangères.

1) À ce point du T.P, la base est sans doute dans un état incohérent suite aux manipulations des TPs 5 et 6. Réinitialiser donc cette base en supprimant toutes ses tables, puis en la recréant grâce au code SQL exporté dans le TP 3.

2) S'assurer que chaque clé étrangère possède les mêmes propriétés (type, taille, signe) que la clé primaire qu'elle référence (faire les éventuels changements nécessaires).

3) Dans l'onglet Structure, grâce au bouton, ajouter un index sur chaque clé étrangère non encore indexée.

Exercice 1 :

Quels sont les champs concernés (argumenter) ?

4) En cliquant sur Vue relationnelle, dans chaque table, relier chaque clé étrangère à la clé primaire qu'elle référence (en laissant pour l'instant RESTRICT comme stratégie de mise à jour et de suppression).

Exercice 2 :

Recopier et expliquer l'une des requêtes générées lors de ces manipulations, ainsi que le message délivré par MySQL lors de la mise à jour et de la suppression d'une clé primaire référencée (par exemple, modifier le numéro de l'enseignant 15 en 25, supprimer l'enseignant n°10).

5) Dans la Vue relationnelle de la table enseignants, définir nom_ens comme colonne descriptive, puis insérer une nouvelle matière en utilisant l'interface de gestion.

Exercice 3 :

Quels changements pouvez-vous constater

• à l'affichage des données de la table matières ?

• lors de l'insertion d'une nouvelle matière ?

6) Dans la Vue relationnelle de la table matieres, appliquer une mise à jour en cascade de la clé étrangère _num_ens.

Exercice 4 :

Recopier et expliquer la requête générée lors de cette manipulation. Quelle est sa conséquence sur la mise à jour d'un numéro d'enseignant (modifier par exemple le numéro de l'enseignant 15 en 25) ?

7) Supprimer l'enseignant n°14, puis le n°10.

Exercice 5 :

Comment s'explique la différence constatée lors de ces suppressions ? Recopier et expliquer le message délivré par MySQL lors de la seconde. Que se passe-t-il si l'on opte pour la stratégie de suppression en cascade et que l'on supprime l'enseignant n°10 ?

8) Tester toutes les stratégies (CASCADE, SET NULL, NO ACTION, RESTRICT) et leurs conséquences sur les données lors d'une mise à jour ou d'une suppression.

Exercice 6 :

Faire une proposition globale pour la gestion des relations de la base complète afin que son intégrité référentielle soit gérée au mieux : quelles stratégies de mise à jour et de suppression proposez-vous pour chaque clé étrangère, et quelle colonne descriptive choisissez-vous pour chaque table ?