Chapitre 5

Insertion de données en langage SQL

Dans les cours précédents, nous avons appris à sélectionner des informations dans une base de données grâce au **langage SQL**. Mais SQL ne s'arrête pas à la sélection de données : il est tout à fait possible d'ajouter, de modifier et de supprimer des données grâce à SQL. Il est également possible de créer des tables et même des bases de données entières.

Pour illustrer ce cours, vous allez utiliser votre propre base de données, déjà manipulée précédemment. Vous pouvez vous y connecter via l'adresse suivante :

`http://nsi.dellasantina.corsica/phpmyadmin/`

Vous pouvez supprimer toutes les tables se trouvant à l'intérieur, afin de faire place nette pour les tables que nous créérons tout au long de ce cours.

5.1 Introduction

Suite à l'ouverture prochaine d'un nouvel hôtel dans la région, vous êtes engagé afin de mettre en place la base de données permettant l'enregistrement de réservations pour cet hôtel.

Pour la conception de la base de données, c'est le stagiaire qui s'y colle, et il propose de créer les 3 tables suivantes :

- **clients**(id *UNSIGNED INT*, nom *VARCHAR(100)*, prenom *VARCHAR(100)*, email *VARCHAR(100)*)
- **chambres**(numero *UNSIGNED INT*, prix *UNSIGNED FLOAT*)
- **reservations**(id *UNSIGNED INT*, #num_chambre *UNSIGNED INT*, #id_client *UNSIGNED INT*, jour *DATE*)

où clé désigne la clé primaire, #clé désigne une clé étrangère, et *TYPE* le type utilisé

Très content de votre stagiaire, vous lui proposez de mettre à l'épreuve sa base de données, en y ajoutant des clients et des réservations, pour vérifier que tout fonctionne correctement.

5.2 Manipulation des tables

5.2.1 Création d'une table

Il nous faut en premier lieu créer les tables dans notre base de données. Pour cela, le langage SQL dispose de la commande bien pratique `CREATE TABLE`. Voici comment créer la table `clients` avec cette commande :

```
CREATE TABLE clients (
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    nom VARCHAR(100) NOT NULL,
    prenom VARCHAR(100) NOT NULL,
    email VARCHAR(100) NULL,
    PRIMARY KEY(id)
)
```

Plusieurs remarques sur la requête ci-dessus :

- le mot clé `UNSIGNED` est placé **après** le type de données `INT`
- le mot clé `NOT NULL` signifie qu'une colonne ne peut accepter de valeurs égales à `NULL`. Pour la clé `email`, supposée ici facultative, on inscrit simplement `NULL`. Par défaut, une colonne est `NOT NULL`.
- la définition de la clé primaire se fait à la fin, grâce au mot-clé `PRIMARY KEY`
- l'attribut `AUTO_INCREMENT` permet d'incrémenter automatiquement les valeurs d'une clé. Chaque client est ainsi identifié de manière unique par un nombre arbitraire, ce qui permet d'éviter par exemple les problèmes d'homonymie.

Question 01 Créer la table `clients` grâce à la requête ci-dessus.

Question 02 Créer de même la table `chambres` en précisant la requête utilisée.

Pour la table `reservations` , il faut préciser les clés étrangères grâce aux mots-clés `FOREIGN KEY` et `REFERENCES` :

```
CREATE TABLE reservations (
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    num_chambre INT UNSIGNED NOT NULL,
    id_client INT UNSIGNED NOT NULL,
    jour DATE NOT NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(num_chambre) REFERENCES chambres(numero),
    FOREIGN KEY(id_client) REFERENCES clients(id)
)
```

Quelques remarques :

- Tout comme la clé primaire, les clés étrangères se définissent « à la fin »
- `FOREIGN KEY` précise le nom de la clé étrangère, et `REFERENCES` précise le nom de la table et la clé associée
- Il est **indispensable** qu'une clé étrangère soit **du même type** que la clé à laquelle elle fait référence

Question 03 Créer la table `reservations` en utilisant la requête ci-dessus.

5.2.2 Modification d'une table

Vos tables sont créées, mais vous faites remarquer au stagiaire que l'hôtel dispose de chambres « luxe » qui possèdent des noms en plus de leurs numéros. Il faut alors modifier la table `chambres` , grâce à la commande `ALTER` :

```
ALTER TABLE chambres
ADD COLUMN nom VARCHAR(100) NULL
AFTER numero
```

Quelques remarques :

- certaines chambres ne possédant pas de nom, on précise que la clé `nom` peut être égale à `NULL` avec le mot-clé idoine
- le mot-clé `AFTER` permet de préciser la place de la colonne dans la table. Par défaut, elle se trouvera à la fin. On précise ici que `nom` sera placé après `numero` . En pratique, cela n'a aucune importance

Question 04 L'hôtel souhaite faire des statistiques sur la provenance de ses clients, majoritairement français. Le stagiaire propose alors de rajouter une colonne `code_postal` à la table `clients` , mais il faut également prendre en compte les clients étrangers, même s'ils sont peu nombreux. Modifier la table `clients` en conséquence en précisant la requête utilisée.

Question 05 Lorsqu'un client réserve une chambre, il peut également réserver un certain nombre de petits-déjeuners (autant que de personnes qui dorment dans la chambre). Par défaut, ce nombre est égal à 0.
Modifier la table `reservations` en conséquence en précisant la requête utilisée.

☞ *le mot-clé `DEFAULT` permet de définir une valeur par défaut : `nom_cle TYPE_DONNEE DEFAULT valeur`*

5.2.3 Vidage et suppression d'une table

Pour vider une table, c'est à dire supprimer toutes les données contenues à l'intérieur, on utilise le mot-clé `TRUNCATE TABLE` :

```
TRUNCATE TABLE nom_table
```

Il est possible de supprimer complètement une table (données et structure) avec le mot-clé `DROP TABLE` :

```
DROP TABLE nom_table
```

▷ ***Attention, MySQL ne demande pas de confirmation !***

5.3 Manipulation de données

5.3.1 Insérer des données

L'insertion de données dans une table se fait avec les mots-clés `INSERT INTO` et `VALUES`.
Supposons que l'on souhaite ajouter la chambre n°42, qui ne possède pas de nom, et dont le tarif est de 50 € la nuit.

```
INSERT INTO chambres
    VALUES (42, NULL, 50)
```

Lorsque la valeur est une chaîne de caractères, il faut l'écrire entre guillemets :

```
INSERT INTO chambres
    VALUES (666, "Chambre de l'enfer", 80)
```

Par défaut, il faut inscrire les valeurs de chacune des colonnes, dans le même ordre que celui défini dans la table.
On peut cependant préciser un ordre différent avec la syntaxe suivante :

```
INSERT INTO chambres (numero, prix, nom)
    VALUES (666, 80, "Chambre de l'enfer")
```

C'est très pratique lorsque certaines colonnes ont des valeurs par défaut ou peuvent contenir des valeurs nulles. On ne renseigne alors que les valeurs qui doivent être non nulles, comme pour notre chambre n°42 :

```
INSERT INTO chambres (numero, prix)
    VALUES (42, 50)
```

Il est également possible d'insérer plusieurs lignes d'un coup, en séparant les n-uplets de valeurs par des virgules :

```
INSERT INTO chambres
    VALUES (42, NULL, 50), (666, "Chambre de l'enfer", 80)
```

Question 06 En plus des chambres 42 et 666 précédentes, insérer les chambres ci-contre dans la table `chambres`, à l'aide de requêtes SQL.

Numéro	Nom	Prix
100	-	60
101	-	60
102	-	60
314	Pi-Room	159
400	Excelsior	300

Question 07 La commande suivante renvoie une erreur. Expliquer pourquoi.

```
INSERT INTO chambres
    VALUES (102, "Chambre avec terrasse", 123.45)
```

Pour ajouter un client, c'est sensiblement identique, à une différence près : la clé `id` est en `AUTO_INCREMENT`, et il n'est donc pas obligatoire de la renseigner. Par exemple, pour ajouter Jean DUPONT, on pourra écrire :

```
INSERT INTO clients
    VALUES (NULL, "Dupont", "Jean", "jean.dupont@supermail.com", 75000)
```

ou encore :

```
INSERT INTO clients (nom, prenom, email, code_postal)
    VALUES ("Dupont", "Jean", "jean.dupont@supermail.com", 75000)
```

Question 08 Donner la requête SQL qui permet d'ajouter à la table `clients` M. ALBERTINI Albert, né le 3 août 1964, habitant Ajaccio, et disposant de l'adresse email `jalbertini@hmail.com`.
Ajouter M. ALBERTINI dans la table `clients` grâce à cette requête.

Question 09 Une des deux requêtes SQL suivantes retourne une erreur, laquelle ? et pourquoi ?

```
INSERT INTO clients (nom, prenom, code_postal)
    VALUES ("Giocanti", "Charles", 20112)
```

```
INSERT INTO clients (nom, email, code_postal)
    VALUES ("Santoni", "a.santoni@supermail.com", 20140)
```

Question 10 Insérer au moins 5 clients dans la table `clients`, dont 1 étranger.

Enfin, on peut ajouter des réservations, mais il faut prendre en compte les clés étrangères. Par exemple, on peut écrire :

```
INSERT INTO reservations (num_chambre, id_client, jour)
    VALUES (1618, 13, "2021-03-14")
```

Cette requête ne fonctionnera pas dans les cas suivants :

- la chambre dont le numéro est 1618 n'existe pas
- le client dont l'id est 13 n'existe pas

En effet, les clés `num_chambre` et `id_client` sont des clés étrangères, associées aux clés primaires `numero` et `id` des tables `chambres` et `clients`. MySQL interdit donc d'ajouter une réservation pour une chambre ou un client qui n'existe pas. C'est **la raison d'être des clés étrangères** et des bases de données relationnelles : **assurer l'intégrité des données**.

Question 11 Ajouter une réservation pour M. ALBERTINI dans la chambre 666 pour le 20 Juin 2021, avec un petit déjeuner.

Question 12 Ajouter plusieurs réservations avec et sans petit déjeuner.

5.3.2 Modifier des données

Une fois des données insérées dans une table, il est possible de les modifier. Cette mise à jour se fait grâce au mot-clé `UPDATE`.
Par exemple, supposons que l'on veuille renommer la chambre 666 en « Chambre du Diable ».
On peut utiliser la requête suivante :

```
UPDATE chambres
    SET nom = "Chambre du Diable"
    WHERE numero = 666
```

- Le mot-clé `UPDATE` est suivi du nom de la table à modifier
- Le mot-clé `SET` permet de préciser la ou les colonnes à modifier
- Le mot-clé `WHERE` définit une condition à remplir pour que la ligne soit mise à jour

Il est possible de modifier plusieurs colonnes en séparant les clés à modifier par des virgules.

```
UPDATE chambres
    SET nom = "Chambre du Diable", prix = 60
    WHERE numero = 666
```

Avec cette requête, on modifie le nom et le prix de la chambre 666.

La condition suivant le `WHERE` n'est pas nécessairement une condition d'égalité, ni une condition portant sur une clé primaire. On peut donc **modifier plusieurs lignes** voire l'ensemble des lignes de la table selon la condition donnée.

Supposons que l'on veuille abaisser à 199 € le prix des chambres dont le tarif est supérieur ou égal à 200 € . On écrira :

```
UPDATE chambres
    SET prix = 199
    WHERE prix >= 200
```

Encore plus fort : imaginons que l'on souhaite augmenter de 10 € le prix des chambres dont le prix est inférieur à 100 € . On peut réaliser cette modification avec la requête suivante :

```
UPDATE chambres
    SET prix = prix + 10
    WHERE prix <= 100
```

Dans l'instruction `SET prix = prix + 10` , le premier mot `prix` fait référence à la colonne à modifier, et le deuxième mot `prix` fait référence aux valeurs de cette colonne. Une chambre dont le tarif est de 50 € passera à 60 € , alors qu'une chambre à 80 € passera à 90 € . Les chambres dont le tarif est strictement supérieur à 100 € ne verront pas leurs tarifs modifiés.

Sans clause `WHERE` , ce sont toutes les lignes de la tables qui sont modifiées :

```
UPDATE chambres
    SET prix = 100
```

Ici, les tarifs de toutes les chambres sont fixés à 100 € .

Question 13 M. ALBERTINI souhaite modifier sa réservation du 20 Juin 2021 : il souhaite la décaler au 26 Juin 2021, et réserver 2 petits déjeuners, car sa femme passera sa nuit à l'hôtel avec lui. Il désire également changer de chambre, le nombre 666 dérangeant quelque peu sa femme. Quelle requête SQL permet de faire cette modification ?

Question 14 À cause de son numéro et de son nom, la chambre 666 n'est jamais réservée par les clients.
Proposer une requête SQL afin de résoudre le problème.

Question 15 Pour attirer plus de clients, l'hôtel fait une réduction de 20% sur les tarifs de toutes les chambres. Quelle requête SQL permet de faire cette modification ?

Un mot sur les clés étrangères

Il n'est pas possible d'effectuer une modification sur une table si cette dernière vient casser l'intégrité de la base de données. Par exemple, si la chambre 666 est réservée une ou plusieurs fois, alors on ne pourra pas changer son numéro, et la commande suivante renverra une erreur :

```
UPDATE chambres
    SET numero = 555
    WHERE numero = 666
```

On pourra cependant changer son nom sans erreur, car contrairement à son numéro, le nom d'une chambre n'est pas clé étrangère dans une quelconque table de la base de données.

Question 16 La requête suivante peut-elle renvoyer une erreur ? Si oui, à quelle(s) condition(s) ?

```
UPDATE reservations
    SET id_client = 14
    WHERE id = 137
```

Question 17 On considère la requête SQL suivante :

```
UPDATE clients
    SET email = NULL
```

Décrire le rôle de cette requête. Peut-elle retourner une erreur ?

5.3.3 Supprimer des données

Pour supprimer des lignes dans une table, on utilise simplement le mot clé `DELETE FROM` , suivi (ou non) d'une clause `WHERE` . Par exemple, pour supprimer la chambre 101, on écrira :

```
DELETE FROM chambres
    WHERE numero = 101
```

Il est possible que cette commande renvoie une erreur si la chambre 101 est réservée. En effet, si la chambre 101 était supprimée, alors les réservations de la chambre 101 « pointeraient » vers une chambre inexistante, ce qui n'est pas permis par MySQL à cause de la clé étrangère (ou plutôt grâce à clé étrangère).
Si la chambre 101 n'est pas réservée, alors la requête ci-dessus serait exécutée sans erreur.

☞ *Il est possible, lors de la définition de la clé étrangère, de préciser l'action à faire lors de la modification ou de la suppression d'une ligne associée à une clé étrangère, avec l'attribut* `ON UPDATE` *ou* `ON DELETE` *. Nous avons déjà vu cela dans le chapitre 2 sur les bases de données. Afin de ne pas compliquer les choses, nous n'en reparlerons pas ici.*

Question 18 On considère la requête SQL suivante :

```
DELETE FROM clients
    WHERE code_postal = 20000
```

Décrire le rôle de cette requête. Peut-elle retourner une erreur ?

Question 19 Mêmes questions avec la requête suivante :

```
DELETE FROM reservations
    WHERE jour >= "2020-01-01"
```

Question 20 Donner une requête qui supprime toutes les réservations de Juin 2020.