TP : Requêtes de mises à jour

Instruction générale : hormis pour les exercices corrigés collectivement, vous ferez valider votre travail par l'enseignant.

1 Mise à jour des tables

On rappelle les commandes concernant la création, la modification et la suppression des tables et leurs attributs.

Création d'une base	`CREATE DATABASE base ;`
Suppression d'une base	`DROP DATABASE base ;`
Création d'une table	`CREATE TABLE table (attributs types conditions) ;`
Ajout de données dans la table	`INSERT INTO table (attribut) VALUES (données);`
Suppression d'une table	`DROP TABLE table ;`
Ajout d'un attribut	`ALTER TABLE table ADD attribut type ;`
Suppression d'un attribut	`ALTER TABLE table DROP COLUMN attribut ;`
Modification d'un attribut	`ALTER TABLE table ALTER COLUMN attribut type ;`
Ajout d'une clé étrangère	`ALTER TABLE table ADD FOREIGN KEY (attribut) REFERENCES tableEtrangere(attribut) ;`

Exemple : reprenons notre exemple de base de données musicale. Notre table `artistes` ne contient que très peu d'attributs, nous allons lui en ajouter. Nous allons ajouter une nationalité à nos artistes. Pour cela, nous allons déclarer un nouvel attribut : `ALTER TABLE artistes ADD nationalite_artiste TEXT`.

Remarques :

- Il est possible d'ajouter une clé étrangère ; pour cela, l'attribut doit être crée avant de pouvoir le déclarer en tant que clé étrangère, il n'est pas possible de faire les deux simultanément ;
- l'ajout d'une clé étrangère à l'aide de la commande `ADD FOREIGN KEY` n'est pas possible sur DB Browser car ce dernier se base sur SQLite qui ne permet pas l'usage de cette commande ; cela doit être fait via l'interface vue dans le premier TP.

Exercice 1. [On prend les mêmes et on recommence]

1. Supprimer l'attribut `Total` de la table `pokemons`.
2. Recréer cet attribut dans la même table. Afficher tous les enregistrements qu'il contient.

2 Mise à jour des données

2.1 Mise à jour interne à la table

On rappelle les commandes concernant l'ajout, la suppression et la mise à jour de données dans une table.

Ajout de données	`INSERT INTO table VALUES (valeur_1,valeur_2,...) ;`
Mise à jour de données	`UPDATE table SET attribut_1=valeur_1,... WHERE condition ;`
Suppression de données	`DELETE FROM table WHERE condition ;`

Remarques :

- Il est possible d'ajouter des données seulement pour certains attributs en les spécifiant à l'aide de la syntaxe suivante : `INSERT INTO table (attribut_1,attribut_2,...) VALUES (valeur_1,valeur_2,...)`.
- Pour supprimer tous les enregistrements sans supprimer la table, il suffit d'utiliser `DELETE FROM table` sans préciser de conditions.

Exemple : Scorpions est un groupe allemand, nous allons mettre à jour sa nationalité dans la table artiste :

```
UPDATE artistes
SET nationalite_artiste="Allemand"
WHERE nom_artiste="Scorpions"
```

Pour les autres, comme nous avons plusieurs groupes britanniques et américains, pour gagner du temps, on peut procéder comme suit :

```
UPDATE artistes
SET nationalite_artiste="Angleterre"
WHERE nom_artiste IN ("Pink Floyd","The Who","Led Zeppelin","Queen") ;
```

et

```
UPDATE artistes
SET nationalite_artiste="États-Unis"
WHERE nom_artiste IN ("Greta Van Fleet","The Doors") ;
```

Exercice 2.

1. L'attaque « snap-trap » a été remplacée par « clap-trap » suite à une mise à jour. Corriger cette erreur.
2. Quel est le type du pokémon Martoni ? Qu'en penser ? Effectuer une recherche dans les autres tables sur ce type.
3. Quel type d'attaque a été remplacé par le type de la question précédente ? Corriger cela.

Il est possible d'utiliser les données déjà présente dans un enregistrement pour mette à jour ce dernier, notamment grâce à des opérations mathématiques.

Exemple : considérons la table et l'enregistrement suivant :

table			
id	attribut1	attribut2	attribut3
1	2	3	

Actuellement, la valeur d'`attribut3` est `NULL`. On va la mettre à jour en lui donnant pour valeur le résultat de l'opération `attribut1`+`attribut2`. Le script suivant permet d'effectuer cela.

```
UPDATE table
SET valeur3=valeur1+valeur2;
```

Exercice 3. L'attribut `Total` de la table `pokemons` était la somme des valeurs des autres attributs représentant les statistiques du pokémon. Mettre à jour la table de façon à recalculer la valeur de l'attribut `Total`.

2.2 Mise à jour à partir d'une autre table

Il est possible d'ajouter des données provenant d'une autre table. Pour cela, on peut utiliser la syntaxe :

```
INSERT INTO table1(attribut1)
SELECT attribut2
FROM table2 ;
```

Exemple : nous allons créer une table `nationalités` afin d'uniformiser cette donnée au sein de la table `artistes`. Elle pourra par ailleurs resservir pour d'autre chose. Créons d'abord la table :

```
CREATE TABLE "nationalités" (
    "id_nationalite" INTEGER NOT NULL DEFAULT 1 UNIQUE,
    "nationalite" TEXT NOT NULL UNIQUE,
    PRIMARY KEY("id_nationalite" AUTOINCREMENT)
);
```

On peut alimenter la table nationalité à l'aide des différentes nationalités de la table `artistes`. Comme il y a beaucoup de redondance au sein de celle-ci, on va utilise la commande `SELECT DISTINCT` afin de ne garder que les résultats différents.

```
INSERT INTO nationalités(nationalite)
SELECT DISTINCT nationalite_artiste
FROM artistes ;
```

Exercice 4.

1. Créer une table `générations` contenant l'attribut `génération`. Quelles sont les contraintes de cet attribut ?
2. Alimenter cette table à l'aide des différents numéros de générations présents dans les autres tables.

Il est aussi possible d'aller récupérer une valeur d'une autre table en passant par un attribut intermédiaire entre les deux tables.

Exemple : Maintenant que la table `nationalités` est créée et alimentée, nous pouvons remplacer dans la table `artistes` l'attribut `nationalite_artiste` par l'identifiant correspondant. Cela permettra d'alléger les tables et augmentera leur intégrité. Pour cela on crée un attribut `id_nationalite` dans la table `artistes` à l'aide de la commande :

```
ALTER TABLE artistes
ADD id_nationalite INTEGER
```

puis on l'alimente à l'aide de la commande mise à jour

```
UPDATE artistes
SET id_nationalite = (
SELECT id_nationalite
FROM nationalités
WHERE artistes.nationalite_artiste = nationalite) ;
```

Remarque : cet exemple permet de constater que l'on peut combiner les différentes commandes à l'intérieure les unes des autres, ici un `SELECT` à l'intérieur d'un `UPDATE`.

Exercice 5.

1. Ajouter à la table `pokemons` les attributs `id_type_1` et `id_type_2`.
2. Mettre à jour ces deux nouveaux attributs de façons à ce qu'ils correspondent aux types des pokémons.
3. Supprimer les attributs `type_1` et `type_2` devenus inutiles.

3 Ressources supplémentaires

— Vidéo sur les requêtes de mise à jour
— Cours du W3 Schools sur le SQL
— Cours du site sqh.sh
— Mémo du site sql.sh
— Plus de fichiers CSV sur les pokémons sont disponibles sur ce Github.