TP4- Introduction au langage SQL

Durée : 2h

1 Introduction

L'objectif est de ce TP est de présenter le langage **SQL** (*Structured Query Language*) qui permet d'interagir avec une base de données.

Même si le langage SQL n'est pas sensible à la casse (présence ou non de majuscules), on prendra l'habitude, pour des question de lisibilité, d'**écrire les commandes du langage SQL en majuscule** (SELECT, INSERT,...).

En SQL, les commentaires sont signalés par -.

L'indentation ou le saut de lignes, contrairement à Python, n'ont pas d'importance. Une requête peut tout à fait prendre une ligne, mais dès qu'elle devient longue, on préfère sauter des lignes et indenter pour gagner en lisibilité.

2 Manipulation d'un table

2.1 Création d'une table

Création d'une table

Pour créer une table, on utilise la commande CREATE TABLE. La syntaxe suivante :

```
CREATE TABLE nom_de_la_table (
    attribut_1 DOMAINE_1,
    ...,
    attribut_p DOMAINE_p,
    PRIMARY KEY(attribut_i),
    FOREIGN KEY(attribut_j) REFERENCES nom_autre_table (attribut_k),
    )
```

permet de créer une table dont :

- le nom est « nom_de_la_table » ;
- les attributs sont « attribut_1 »,..., « attribut_p » et ont pour domaine respectif « DOMAINE_1 »,..., « DOMAINE_p »,
- le i-ème attribut « attribut_i » est une clé primaire,
- le j-ème attribut « attribut_j » est une clé étrangère qui référence le k-ième attribut d'un autre table dont le nom est « nom_autre_table »,

On considère les trois tables suivantes :

Propriétaires

Id_propriétaire	Nom	Prénom	Téléphone
H1	Ghilder	Mateo	0668543452
H2	Troccovee	Anne	0675463309
H3	Chasoeler	Philip	0667240908
H4	Bobont	Charline	0638899821
H5	Ricciardo	Louis	0619798669

Chats

Id_chat	Nom	Propriétaire	Pelage
C1	Aramis	H3	nb1
C2	Lambda	H3	cb1
C3	Gerda	H4	nb2
C4	Lili	H5	tm1
C5	Barbaque	H1	r1
C6	Reblochon	H2	nb2

Pelages		
Id_pelage	**Poil**	**Couleurs**
r1	court	roux
nb1	court	noir à tâches blanches
cb1	court	crême et blanc
nb2	court	blanc à tâches noires
tm1	mi-long	noir, brun et blanc

Exemple 1

Pour créer la table `Pelages`*, on utilise la commande :*

```
CREATE TABLE Pelages (
    Id_pelage TEXT,
    Poil TEXT,
    Couleurs TEXT,
    PRIMARY KEY(id_pelage),
    )
```

► Donner la commande permettant de créer la table `Propriétaires`.

► Donner la commande permettant de créer la table `Chats`.

2.2 Modification d'une table

Pour le moment, les tables créées à la partie précédente sont vides. Il va falloir y ajouter les lignes souhaitées.

Ajout de lignes dans une table

La syntaxe suivante :

```
INSERT INTO nom_de_la_table VALUES (valeur_1,...,valeur_p)
```

permet d'ajouter à la table « nom_de_la_table » un ligne dont la valeur de « attribut_1 » est « valeur_1 »,...
Il est important de noter que :

- Les éléments sont saisis dans l'ordre des colonnes, il ne faut pas en oublier !
- Il ne faut pas oublier les guillemets autour des valeurs de type TEXT !

Exemple 2

Pour ajouter une ligne à la table `Propriétaires`, on utilise la commande :

```
INSERT INTO Propriétaires VALUES ("H1","Ghilder","Mateo",0668543452)
```

► Donner la commande permettant d'ajouter la première ligne à la table `Chats`.

► Rentrer toutes les lignes des trois tables.

Suppression de lignes dans une table

Pour supprimer des lignes vérifiant une certaine condition, on utilise la commande :

```
DELETE FROM nom_de_la_table WHERE condition
```

Cette commande supprime les lignes de la table « nom_de_la_table » vérifiant la condition « condition ».
La condition s'exprime à l'aide d'opérateurs de comparaison :

- A = B : A est égal à B ;
- A <> B : A est différent de B ;
- A > B et A < B : A est supérieur/inférieur à B ;
- A >= B et A =< B : A est supérieur/inférieur ou égal à B

et d'opérateurs logiques :

- AND : et ;
- OR : ou ;
- NO : négation.

Exemple 3

Si on souhaite supprimer de la liste `Chats` toutes les lignes où le propriétaire est `H3` ou `H1` on utilise la commande :

```
DELETE FROM Chats WHERE (Propriétaire = "H1") OR (Propriétaire = "H3")
```

► Donner la commande permettant de supprimer de la table `Pelages` toutes les lignes où le poil n'est pas court.

Suppression d'une table

La commande :

```
DELETE FROM nom_de_la_table
```

supprime la table « nom_de_la_table ».

Modification de données dans une table

La syntaxe

```
UPDATE nom_de_la_table SET attribut ="new_valeur " WHERE condition
```

permet de remplacer la valeur de l'attribut `attribut` par la valeur `new_valeur` dans toutes les lignes vérifiant la condition `condition`.

Exemple 4

Si Philip décide de donner ses chats à Mateo, il faut mettre à jour la base de données à l'aide de la commande :

```
UPDATE Chats SET Propriétaire ="H1" WHERE Propriétaire="H3"
```

► Le numéro de téléphone d'Anne est erroné : le dernier chiffre devrait être un 8. Donner la commande permettant de corriger cette erreur.

2.3 Sélection de données dans une table

Sélection de données dans une table (Projection)

La **projection** consiste à sélectionner seulement une partie des attributs. Cela se fait par la commande

```
SELECT attribut_1,...,attribut_r FROM nom_de_la_table
```

où :

- « nom_de_la_table » est le nom de la table ;
- les attributs « attribut_1 »,..., « attribut_r » sont les attributs que l'on souhaite sélectionner.

La commande :

```
SELECT * FROM nom_de_la_table
```

permet de sélectionner tous les attributs de la table.

► Écrire la commande permettant de sélectionner la table `Propriétaires`.

► Écrire la commande permettant de sélectionner l'attribut `Nom` de la table `Chats`.

Sélection de données dans une table (Sélection ou restriction)

Une **restriction** dans une table de données est une condition (de type vrai ou faux) qui porte sur un ou plusieurs des attributs de la relation et qui va sélectionner les lignes pour lesquelles la condition est vérifiée.
La commande

```
SELECT attribut_1,...,attribut_r FROM nom_de_la_table WHERE condition
```

sélectionne les attributs les attributs « attribut_1 »,..., « attribut_r » des lignes pour lesquelles la condition `condition` est satisfaite.
La condition s'exprime avec les mêmes opérateurs de comparaison que pour la suppression de lignes (2.2).

Exemple 5

Si on souhaite sélectionner de la liste `Chats` les `Couleurs` correspondant à des `Poils` courts on utilise la commande :

```
SELECT Couleurs FROM Chats WHERE Poil = "court"
```

► Donner la commande permettant d'afficher le `Nom` des chats dont le propriétaire est Philip (H3).

3 Jointure

La jointure est le concept fondamental de l'algèbre relationnelle. Nous avons vu que les clés étrangères permettent de lier des tables entre elles. Mais nous ne savons pas encore comment exploiter ces liaisons. C'est justement le but de l'opération de jointure.

La jointure sert à coller les deux tables pour faire apparaître l'information sur une seule table.

Jointure

On considère deux tables `table_1` et `table_2` et on suppose que `table_1`, l'attribut `fk` est une clé étrangère qui référence une clé primaire `pk` pour de `table_2`. La **jointure** de `table_1` et `table_2` selon la condition `fk` = `pk` est la table qui contient le même nombre de lignes que `table_1` et les attributs de `table_1` et de `table_2`. Chaque ligne est construite en commençant par la ligne de la `table_1` , puis en ajoutant à sa suite la ligne de la `table_2` caractérisée par la valeur de sa clé.

```
SELECT * FROM table_1 INNER JOIN table_2 ON (table_1.fk = table_2.pk)
```

Exemple 6

La commande :

```
SELECT * FROM Chats INNER JOIN Propriétaires
ON (Chats.Propriétaire = Propriétaires.Id_propriétaire)
```

affiche :

Id_chat	***Nom***	***Propriétaire***	***Pelage***	***Id_propriétaire***	***Nom***	***Prénom***	***Téléphone***
C1	*Aramis*	H3	*nb1*	H3	*Chasoeler*	*Philip*	0667240908
C2	*Lambda*	H3	*cb1*	H3	*Chasoeler*	*Philip*	0667240908
C3	*Gerda*	H4	*nb2*	H4	*Bobont*	*Charline*	0638899821
C4	*Lili*	H5	*tm1*	H5	*Ricciardo*	*Louis*	0619798669
C5	*Barbaque*	H1	*r1*	H1	*Ghilder*	*Mateo*	0668543452
C6	*Reblochon*	H2	*nb2*	H2	*Troccovee*	*Anne*	0675463309

► Écrire la commande permettant d'afficher dans une même table les attributs des chats et de leur pelage.