Insertion des données dans une BD

La commande INSERT

Insertion des enregistrements dans une table

- Une fois les tables sont créés , la première chose à faire c'est de saisir des données. Une occurrence d'une table on l'appel **enregistrements** ou **lignes** de table.

- La commande utilisée pour ce faire c'est la commande **INSERT**

La commande INSERT

- **Syntaxe 1:**

```
INSERT INTO
nomDeTable(champ1,champ2,champ3,…)
VALUES(valeur1, valeur2, valeur3,…)
```

Rq:

- ✓ Les champs et les valeurs doivent être séparés par une virgule
- ✓ **Le nombre de valeurs à insérer doit correspondre au nombre de champs**
- ✓ **Le type des valeurs doivent se correspondre avec les types des champs.**

La commande INSERT

- Si on va insérer des données dans tous les champs de la table, dans ce cas ce n'est pas la peigne de mentionner les champs.

  Dans ce cas on peut écrire:

```
INSERT INTO nomDeTable VALUES(valeur1, valeur2, valeur3,…)
```

Aucun champ n'est mentionné

Commande INSERT

- Exemple 1:

```
Insert into stagiaires values (1000,'ALAMI','AHMED',
'2000-06-12')
```

- Exemple 2:

```
Insert into stagiaires (Num,nom,prenom,DateNaiss)
values (1000,'ALAMI','AHMED','2000-06-12')
```

La commande INSERT

- Si on veut insérer plusieurs enregistrements à la fois, on peut écrire :

```
INSERT INTO NomTable VALUES (val1,val2,val3) ,
(val11,val22,val33) ,  (val111,val222,val333)
```

- Avec cette syntaxe, on va insérer trois enregistrements dans un seul ordre INSERT

Interroger une BD

Commande SELECT

La commande SELECT

- L'interrogation de la base de données veut dire le **filtrage** des données ou la **recherche**. Cette manipulation est faite grâce à la commande **SELECT**.

- Select est utilisé donc pour l'affichage des données après filtrage. c'est la commande la plus complexe de SQL, elle sert à créer des requêtes pour récupérer les données dans la/les tables.

La commande SELECT

- **Syntaxe**:

```
SELECT  champ1,champ2 ,… FROM NomTable;
```

- Les champs doivent être séparés par des virgules.

- **Remarque** : Les champs sont retournés dans l'ordre spécifié après la clause SELECT et non pas dans l'ordre de leur création dans la table.

La commande SELECT

- Exemple:

```
SELECT nomStagiaire, prenomStagiaire FROM stagiaires
```

Le résultat de la requête SQL dessus est comme suit:

nomstagiaire	prenomstagiaire
ALAMI	mohamed
TAHIRI	ALI
madani	karima
HAMOUTI	laila

Commande SELECT

- Si on veut récupérer les données de tous les champs dans tous les enregistrements, on remplace les noms des champs par une étoile *, on écrit dans ce cas:

```
SELECT * FROM NomTable;
```

Commande SELECT: filtrage

- On peut indiquer dans la commande SELECT des **conditions de sélection**. En utilisant la clause **WHERE**.

- Syntaxe:

```
SELECT * FROM nomTable WHERE condition
```

Où condition est une condition qui est appliquée sur un ou plusieurs champs

Commande SELECT: la clause WHERE

La condition peut contenir:

1. **Les opérateurs de comparaison:**

   Ces opérateurs sont (=, >,>=,<,<=,!=)

- Exemple:

```
Select * from produits where prix_unitaire>=200
```

Commande SELECT: la clause WHERE

La condition peut contenir:

2. **Les opérateurs logiques:**

Ces opérateurs sont: (AND, OR, NOT)

Exemple:

```
Select * from clients where nom='Mohamed' AND ville='Oujda'
```

Commande SELECT: la clause WHERE

- La condition peut contenir:

3. **La clause IN/NOT IN:**

Cette clause est utilisée lorsque nous avons un champ qui appartient à une liste de valeurs.

Exemple:

```
Select * from clients where ville IN
('Oujda','Rabat','Tanger')
```

Commande SELECT: la clause WHERE

La condition peut contenir:

4. **La clause BETEWEEN/ NOT BETWEEN**

Cette clause est utilisée lorsqu'on veut que la valeur d'un champ soit comprit entre deux valeurs.

Exemple:

```
Select * from produits where prix BETWEEN 300
AND 800
```

Commande SELECT: la clause WHERE

- La condition peut contenir:
  5. **La clause LIKE / NOT LIKE:**

  Cette clause est utilisée lorsqu'on veut qu'un champ soit 'comme' une certaine forme.

```
Select * from clients where nom like 's%'
 --afficher tous les clients dont le nom commence par s

Select * from clients where nom like '%e'
--afficher tous les clients dont le nom se termine par e
```

Rq: on utilise la clause LIKE de la même manière qu'on l'utilise avec la clause Check (voir le cours)

Commande SELECT: la clause WHERE et l' expression REGEXP

- On peut effectuer une recherche avancée avec LIKE, mais cette clause reste limitée et ne peut pas faire le tout.

- **REGEXP** permet de faire un filtrage plus précis. Cette clause permet d'utiliser les expressions régulières dans MySQL.

Commande SELECT: l'expression REGEXP

- Syntaxe:

```
Select  * from   table where champ    REGEXP  'expression'
```

Cette requête va récupérer tous les enregistrements de la table dont le champ correspond à l'expression spécifiée.

L'expression REGEXP

La syntaxe des expressions, sont les mêmes que celle qu'on a vu en **javascript**.

1. **^ : correspond au début d'une chaine**

```
Select * from clients where ville regexp '^A';
```

Le champ ville doit commencer par A

L'expression REGEXP

2. **$: correspond à la fin d'une chaine de caractère**

```
Select * from clients where ville regexp 'e$';
```

Le champ ville doit se terminer par e

L'expression REGEXP

3. **. : Le point correspond à n'importe quel caractère**
4. **a*:** Le caractère **a apparait Zéro ou plusieurs fois**.

```
Select * from clients where CodeCLient regexp
'^A.*e$';
```

Le code du client commence par **A** suivi d'un **caractère(ou aucun)** et se termine par **e**

L'expression REGEXP

5. **a+** : Le caractère **a apparait une ou plusieurs fois**.

```
Select * from Clients where ville regexp '^As+';
```

La ville du client commence par **A** suivi de un ou plusieurs **s**.

L'expression REGEXP

6. **a?** : Le caractère a apparait zéro ou une seule fois.
7. **x| y** : signifie soit x soit y
8. **a{n}** : correspond à **n** occurrences de **a**
9. **a{m,n}** : le caractère **a** apparait **m** fois minimum et **n** fois maximum.
10. **a{n,}** : Le caractère **a** apparait **n** fois minimum.

L'expression REGEXP

- **[0-9]** : correspond à n'importe quel chiffre décimal
- **[a-z]:** correspond à n'importe quelle lettre alphabétique
- **[acdg]:** correspond exactement à l'un des caractères a ou c ou d ou g

Rq: Il existe d'autres symboles (voir le fichier correspondant aux expressions régulières)

Filtrer les enregistrements : IS NULL/IS NOT NULL

- Un champ avec une valeur **NULL** est un champ sans valeur.
- Un champ avec une valeur NULL est un champ qui a été laissé vide lors de la création de l'enregistrement.
- Pour vérifier qu'un champ contient une valeur NULL ou non on utilise la syntaxe suivante :

```
SELECT Colonnes FROM NomTable WHERE colonneX
IS NULL/IS NOT NULL ;
```

La commande SELECT: la clause DISTINCT

- Le mot **distinct** permet de supprimer les doublons (s'il y'a des lignes qui se répètent) celle-ci va laisser une seule.
- Exemple: si on exécute la requête suivante elle peut nous donner des doublons: **Select nomStg from stagiaires**
- Pour éliminer les doublons on écrira:

```
Select DISTINCT nomStg from stagiaires
```

La commande SELECT: La clause ORDER BY

- La clause ORDER BY Permet de **trier** les enregistrements sélectionnés selon un champ, dans l'ordre ***croissant*** ou ***décroissant***.

```
Select * from NomTable ORDER BY champ DESC
```

- DESC c'est décroissant et ASC c'est croissant
- Exemple:

```
Select  *  from  clients  ORDER BY pays desc
```

Rq: l'ordre est par défaut ASC