Ch6 : Manipulation d'une BD

1. Introduction

La manipulation de données consiste à effectuer les opérations suivantes sur les tables de la Base de données :

- la mise à jour de données : insertion, modification ou suppression de lignes d'une table
- consultation, ou recherche de lignes existantes dans une ou plusieurs tables

Comme pour la définition des données, la manipulation de données peut être effectuée selon deux modes :

2. Manipulation en mode commande :

2.1. Mise à jour des tables :

La mise à jour des tables d'une base de données se fait à l'aide de 3 commandes : la commande **d'insertion,** la commande de **modification** commande de **suppression** de lignes.

a. Insertion de lignes :

L'insertion de nouvelles lignes dans une table dont la structure a été déjà créée dans la base se fait en utilisant la commande **INSERT INTO.**

La forme générale de cette commande est la suivante :

```
INSERT INTO nom_table [(colonne1, colonne2,… colonne n)]
VALUES (valeur1, valeur2,… ,valeur n)
```

• **[(colonne1, colonne2,… colonne n)]** (données par la commande **CREATE TABLE**) **:** sert à préciser la liste des colonnes qui peut être omise dans le où l'opération d'insertion concerne toutes les colonnes de la table et dans ce cas l'ordre des valeurs fournies par **(valeur1, … ,valeur n)** doit être les mêmes que les colonnes.

• Pour que l'opération d'insertion puisse être exécutée, les conditions suivantes doivent être respectées :

- **R 1.** Les types des données de ***liste_valeur*** doivent être compatibles avec ceux colonnes de la table,
- **R 2. Unicité** des lignes (contrainte de clé primaire),
- **R 3.** Caractère **obligatoire** associé à une colonne (clause NOT NULL),
- **R 4. Existence** de la valeur dans une autre table quand il s'agit d'une clé étrangère (contrainte d'intégrité référentielle)
- **R 5.** Vérification d'une condition de validité (contrainte valeur : clause **CHECK**).

EXEMPLE :

Table client							
Code	**Nom**	**Prénom**	**Adresse**	**Téléphone**	**E-mail**	**Chiff. Aff.**	**Cumul**
S0010	Gloulou	Rania	Sousse	66536658	Rania.gloulou.gnet@gnet.tn	15679.355	123765.540

METHODE 1 :

```
INSERT INTO client (code_cl, nom_cl, pren_cl, adr_cl, tel_cl, email_cl, chif_aff, cumul_ch_aff)
VALUES ('S0010', 'Gloulou', 'Rania','Sousse','66536658','Rania.gloulou.gnet@gnet.tn', 15679.355, 123765.540)
```

METHODE 2 :

```
INSERT INTO client
VALUES ('S0010', 'Gloulou', 'Rania' ,'Sousse','66536658','Rania.gloulou.gnet@gnet.tn', 15679.355, 123765.540)
```

b. modification des lignes :

La modification de contenu d'une colonne d'une ou de plusieurs lignes d'une table se fait en utilisant la commande **UPDATE.**

La forme générale de cette commande est la suivante :

```
UPDATE nom_table
SET colonne1= expression1 [, colonne2= expression2, …, colonne_n= expression_n]
[WHERE condition]
```

☞ Les valeurs des colonnes identifiées par ***colonne_i,*** sont modifiées dans toutes les lignes vérifiant la condition de la clause **WHERE**. En cas où cette dernière est absente, toutes les lignes de la table sont mises à jour.

☞ Les paramètres donnés par ***expression,*** remplace les anciennes valeurs de la colonne identifiée par ***colonne_i***

☞ Pour que l'opération de mise à jours puisse être exécutée, il faut respecter les mêmes conditions qui ont étés indiqués pour l'ordre **INSERT**.

EXEMPLE : modifier **l'adresse** du client dont le code = S0010 à 'TUNIS'

```
UPDATE client
SET adr_client= 'TUNIS'
WHERE code_cl='S0010'
```

b. suppression de lignes :

La suppression d'une ou de plusieurs lignes d'une table se fait en utilisant la commande **DELETE FROM.**
La forme générale de cette commande est la suivante :

```
DELETE FROM  nom_table
 [WHERE condition]
```

☞ Le paramètre ***condition*** qui apparaît dans la clause **WHERE** sert à indiquer la condition qui doit être vérifiée par les lignes à supprimer. Si elle est absente, toutes les lignes de la table concernée seront supprimées.

☞ En cas où la suppression ne respecte pas une contrainte d'intégrité référentielle, elle ne peut pas aboutir.

☞ La suppression d'une ligne appartenant à une table donnée peut entraîner la suppression d'autres lignes appartenant à d'autres tables lorsqu'il existe de contraintes d'intégrité référentielles de suppression en cascade : utilisation de 1a clause ON DELETE CASCADE dans la définition des clés étrangères.

EXEMPLE : supprimer le client dont le code = S0010 de la table client

```
DELETE FROM client
WHERE code_cl='S0010'
```

2.2. Recherche de données : requêtes

La recherche de données peut être réalisée en faisant référence à une ou à plusieurs tables. Elle peut se référer à une partie ou à la totalité des colonnes des tables concernées tout en étant, éventuellement, conditionnelle. Une recherche peut consister à effectuer : une **projection** sur certaines colonnes d'une table, une **sélection** sur certaines lignes d'une table, une **jointure** sur 2 tables ou Toute **combinaison** de projection, sélection et jointure.

A- Recherche de colonnes à partir d'une table : PROJECTION

La **projection**, est un sous-ensemble des colonnes d'une seule table. Le résultat de la projection doit comporter au moins une colonne de la table.

La forme générale de la commande de projection est la suivante :

```
SELECT [DISTINCT] */ colonne1 ['legende1'], colonne2 ['legende2'], …
FROM nom_table
```

☞ **colonne1, colonne2, … :** la liste des colonnes, que l'on veut l'afficher.

☞ Le symbole * : désigne tous les colonnes de la table.

☞ **DISTINCT** sert à éliminer les lignes en double dans le résultat si la clé primaire ne figure pas dans la liste des colonnes à afficher.

☞ **nom_table** : se référer à la table concernée.

☞ Le résultat de **SELECT** est une nouvelle table.

☞ par défaut les colonnes de la table résultat portent les mêmes noms que ceux de la table de départ. Il est possible de les donner des autres noms, dans ce cas donner une légende juste après le nom de la colonne. Cette entête est généralement appelée **ALIAS.**

EXEMPLES

Exemple 1 : Donner les codes, noms et prénoms de tous les clients.

```
SELECT code_cl, Nom_cl, Prénom_cl
FROM Client
```

Exemple 2 : Donner les caractéristiques de tous les clients.

```
SELECT *
FROM Client
```

Exemple 3 : Donner les codes, noms et prénoms de tous les clients. Au moment de l'affichage, les entêtes des colonnes doivent être respectivement 'Code du Client', 'Nom du Client' et 'Prénom du Client'.

```
SELECT code_cl 'Code Client', Nom_cl 'Nom Client', Pren_cl 'Prénom Client'
FROM Client
```

Exemple 4 :

Donner les chiffres d'affaires de l'année en cours de tous les clients.

```
SELECT Chif_Aff
FROM Client
```

La colonne relative aux chiffres d'affaires de l'année en cours sera affichée. En cas où plusieurs clients ont le même montant, on aura plusieurs lignes affichées avec le même montant.

Exemple 5 :

Donner les chiffres d'affaires de l'année en cours de tous les clients. L'affichage d'un montant doit se faire une seule fois en cas d'égalité de certains montants.

```
SELECT DISTINCT chif_Aff
FROM Client
```

Exemple 6 :

Donner la liste des produits et pour chacun calculer la valeur du stock.

```
SELECT des_art 'Désignation Produit', PU*Qte_stock 'Valeur Stock'
FROM Article;
```

Remarque

La colonne résultat, relative à la **valeur du stock**, sera calculée à partir du prix unitaire et la quantité en stock.

B- Recherche de lignes à partir d'une table : SELECTION

La **sélection** ne concerne qu'une seule table de la BD et la différence par rapport à l'opération de projection c'est que le résultat est composé d'un sous-ensemble de lignes de la table.

La forme générale de la commande de sélection est la suivante :

```
SELECT [DISTINCT] */ colonne1 ['legende1'], colonne2 ['legende2'], …
FROM nom_table
WERE condition
```

- ✓ Le paramètre **condition** sert à préciser le critère (ou prédicat) qui doit être vérifié par les lignes à afficher. Ce prédicat est donné sous la forme d'une expression logique.
- ✓ Si le résultat de l'évaluation de l'expression logique est vrai, pour une ligne sera partie du résultat.

Dans l'expression logique, on peut utiliser en particulier :

1. Les opérateurs de comparaison : **=, >, <, >=, <= , <>**
2. L'opérateur **BETWEEN** pour les intervalles de valeur;
3. L'opérateur **IN** pour les listes de valeurs.
4. L'opérateur **IS NULL** et **IS NOT NULL** pour les valeurs indéterminées.
5. L'opérateur **LIKE** pour filtrer une chaîne de caractères
6. Les opérateurs logiques : **AND, OR, NOT**

EXEMPLES

Exemple 1 :

Donner la liste des clients qui ont atteint ou dépassé un chiffre d'affaire de 50000 dinars de l'année en cours.

```
SELECT *
FROM client
WHERE Chif_aff >=50000
```

Seules les lignes qui vérifient la condition indiquée feront parti du résultat.

Exemple 2 :

On demande la même liste que l'exemple précédent sauf qu'on ne s'intéresse qu'aux colonnes relatives aux noms et aux prénoms

```
SELECT nom_client 'Nom', prenom_client 'Prénom'
FROM client
WHERE Chif_aff >=50000
```

Exemple 3 :

Donner la liste des clients qui ont un chiffre d'affaire entre 100000 et 50000 dinars de l'année en cours.

```
SELECT *
FROM client
WHERE Chif_aff BETWEEN 10000 AND50000
```

Exemple 4 :

Donner la liste des clients qui ont un chiffre d'affaire égale à 100000, 200000 ou 50000 dinars de l'année en cours.

```
ELECT *
FROM client
WHERE Chif_aff IN (10000, 20000, 5000)
```

Exemple 5 :

Donner la liste des clients qui ont un nom commence par 'A' et une adresse nulle ou le troisième caractère est 'U'.

```
SELECT    *
FROM client
WHERE   nom_cl LIKE 'A%' AND (adr_cl IS NULL OR adr_cli LIKE '_ _U%'
```

Remarque :

- ✓ Le caractère **%** utilisé avec l'opérateur **LIKE** sert à remplacer une liste de caractères, y compris le vide.
- ✓ Le caractère **_** utilisé avec l'opérateur **LIKE** sert à remplacer un caractère, pour remplacer un nombre de caractères (n) , il suffit de mettre (n) fois successives le caractères _

C- Recherche de lignes à partir de plusieurs tables : JOINTURE

Dans certaines situations, l'utilisateur a besoin de données provenant de plusieurs tables différentes de la BD. Il s'agit dans ce cas d'une opération de **jointure.**

La forme générale de la commande de **Jointure** est la suivante :

```
SELECT [DISTINCT] */ colonne1 ['legende1'], colonne2 ['legende2'], …
FROM nom_table1 [ AS alias1] ,  nom_table2 [AS alias2] , …
WHERE condition
```

- ☞ Un **ALIAS** permet de donner un nom synonyme, abrégé, à une table. Il permet d'alléger l'écriture de la commande SELECT en cas d'ambiguïté.
- ☞ Le paramètre **CONDITION** sert, particulièrement, à préciser la condition de jointure. Cette condition ne doit pas être confondue avec celle indiquée pour l'opération relationnelle de sélection.
- ☞ La **CONDITION** de jointure doit porter sur les colonnes en commun aux tables utilisées pour l'opération de jointure (voir contrainte d'intégrité référentielle : **clé primaire - clé étrangère**).

Exemple 1

Donner la date de chaque commande ainsi que le nom et le prénom du client qui a fait la commande.

```
SELECT Nom_cl, Pren_cl, dte_com
FROM Client as CL, Commande as COM
WHERE CL.code_cl = COM.code_cl
```

Remarques

- Si un même client peut faire plusieurs commandes, le même jour, et qu'on ne veut pas avoir de doublons dans le résultat, la commande devient :

```
SELECT DISTINCT Nom_cl, Pren_cl, dte_com
FROM Client as CL, Commande as COM
WHERE CL.code_cl = COM.code_cl
```

Exemple 2 :

Donner le numéro et la date de chaque commande ainsi que les quantités commandées et les désignations des articles correspondants.

```
SELECT C.num_com, dte_com, Qte_com, des_art
FROM Commande as C, détail_commande as D, article as A
WHERE (C.num_com=D.num_com) AND (D.code_art=A.code_art)
```

Remarque

Les relations ont été utilisées pour répondre à ce besoin. Deux opérations de **Jointure** seront donc exécutées : la 1ière entre les tables **Commande** et **détail_commande** et la **seconde** entre le résultat obtenu et la table **Article**.

Exemple 3

Même besoin que l'exemple précédent sauf que les commandes qui nous intéressent sont celles qui sont relatives à des produits dont la quantité en stock, est suffisante.

```
SELIECT C.num_com, date_com, Qte_com, des_art
FROM Commande as C, detail_coirunande as D, article as A.
WHERE (Qte_stock <= Qte_com AND C.num_com = D.num_com) AND (D.code_art = A.code art)
```

Exemple 4 :

Donner la liste des clients qui ont atteint les mêmes montants des chiffres d'affaires de l'année en cours.

```
SELECT DISTINCT X,Nom_cl,  X.Prénom_cl, X.Chif _Aff
FROM Client As X, Client As Y
WHERE (X.Chiff_Aff =Y.Chiffre Affaires Année encours)
```

Remarque

Dans cet exemple nous avons utilisé deux fois la même table. Il s'agit d'une opération d'auto-jointure. Dans ce cas l'utilisation d'un **alias** dans la commande SELECT devient indispensable.

Exemple 5 :

Donner les clients qui ont atteint le même montant des chiffres d'affaires de du client « Mohamed ben Saleh" .

```
SELECT X,Nom_cl,  X.Prénom_cl, X.Chif _Aff
FROM Client As X
WHERE X.code_cl<>Z.code cl and
        X.Chiff_Aff = ( SELECT Chif _Aff
                        FROM Client As Z
                        WHERE Z,Nom_cl ="Ben Saleh" and  Z.Prénom_cl="Mohamed")
```

D-Recherche de données avec tri :

Certaines requêtes ont besoin de rechercher des données de la Base et d'avoir résultat qui soit trié selon un ordre croissant ou décroissant des valeurs d'une ou de plusieurs colonnes. La forme générale de la commande de recherche est la suivante :

```
SELECT [DISTINCT] */ colonne1 ['legende1'], colonne2 ['legende2'], …
FROM nom_table1 [alias1] [ ,  nom_table2 [alias2] , …]
[WERE condition]
ORDER BY coli/i [Asc/Desc], [colj/j [Asc/Desc],…]
```

La clause **ORDER BY** sert à exiger le classement (ou le tri) du résultat. Le classement peut se faire selon un ordre croissant (ASC) ou décroissant (DESC) des valeurs d'une ou de plusieurs colonnes. Par défaut l'ordre est ASC.

Coli : est un nom du champ parmi les propriétés à afficher avec SELECT

Exemple 1 : afficher les clients triés selon dans l'ordre alphabétique de leur nom

```
SELECT  *
FROM Client
ORDER BY nom_cl ASC
SELECT  *
FROM Client
ORDER BY nom_cl
```

Exemple 2 :

Donner le numéro et la date de chaque commande ainsi que les quantités commandées et les désignations des articles correspondants en triant le résultat selon les dates dans l'ordre décroissant :

```
SELECT C.num_com, dte_com, Qte_com, des_art
FROM Commande C, détail_commande D, article A
WHERE (C.num_com=D.num_com) AND (D.code_art=A.code_art)
ORDER BY  2  DESC
```

E- Requête de calcul :

***calcul simple :**

Exemple1 : pour une liste des notes calculer les moyennes de chaque élève en informatique :

```
SELECT  code_e, (nc+ns*2)/3 'moyenne'
FROM note
WHERE code_m = "info"
GROUPE BY code_e
```

***calcul avec des fonctions agrégats :**

Le langage SQL prévoit des fonctions agrégats qui nous permettent de faire du calcul au niveau des requêtes :

- **COUT** : permet de compter le nombre de lignes résultats obtenues par la commande SELECT

Exemple1 : afficher le nombre des clients

```
SELECT Count(*)
```

```
FROM Client
```

- **SUM** : permet de faire la somme des valeurs d'une colonne dont le type de données est numérique

Exemple2 : afficher les montants des chiffres d'affaires des clients de « Mahdia »

```
SELECT Sum(chif_Aff)
FROM Client
WHERE adr_cl = "Mahdia"
```

- **MIN** : permet d'afficher la valeur minimale
- **MAX** : permet d'afficher la valeur maximale
- **AVG** : permet d'afficher la valeur moyenne arithmétique

Exemple3 : afficher la note de synthèse minimale, la note de synthèse maximale et la moyenne de ses notes en informatique :

```
SELECT E.* , Min(note_e) 'moyenne minimale', Avg(note_e) 'moyenne de la classe'
FROM Eleve  as E, note as N
WHERE N.code_m = "info"  and N. nc >= (S
```

Exemple4 : afficher les élèves qui ont une note de synthèse supérieure à la moyenne de la classe en informatique :

```
SELECT E.*
FROM Eleve  as E, note as N
WHERE N.code_m = "info"  and E.code_e= N.code_e
        and N. nc >= (SELECT nc
                      FROM note
                      WHERE code_m = "info"  )
```

3. Manipulation en mode assisté :

3.1. Mise à jour des tables :

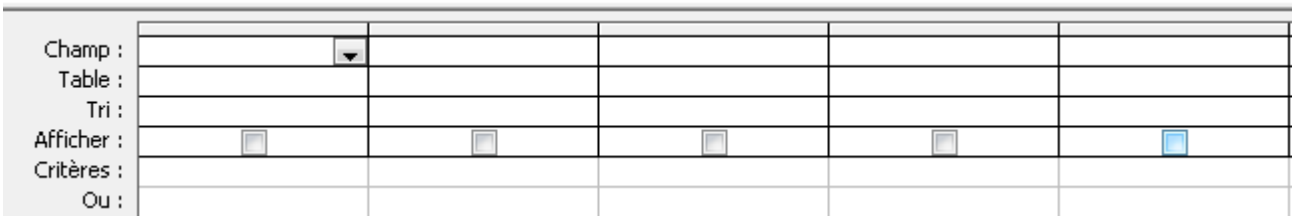
a. Insertion de lignes :

b. modification des lignes :

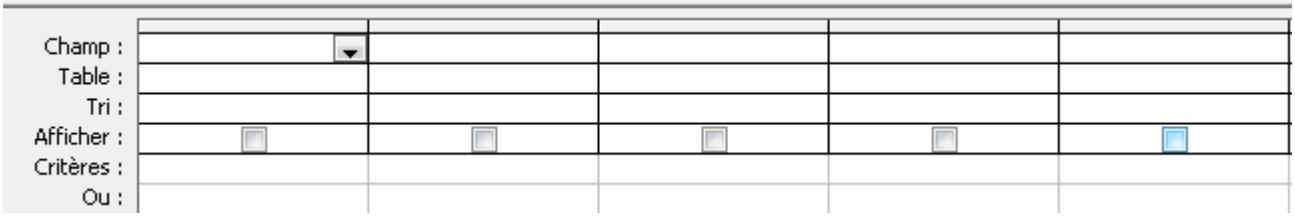
b. suppression de lignes :

3.2. Recherche de données : requêtes

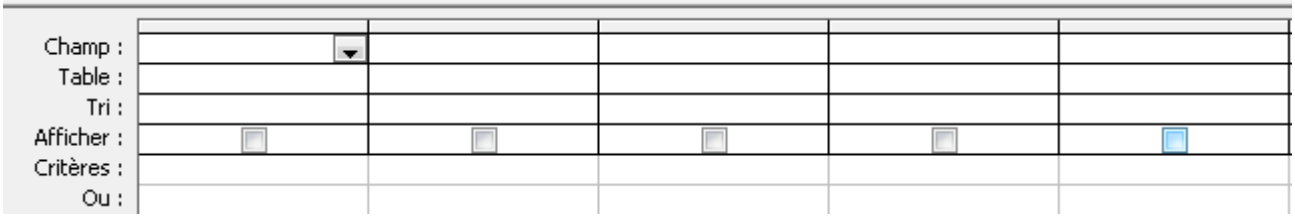
R1. ……



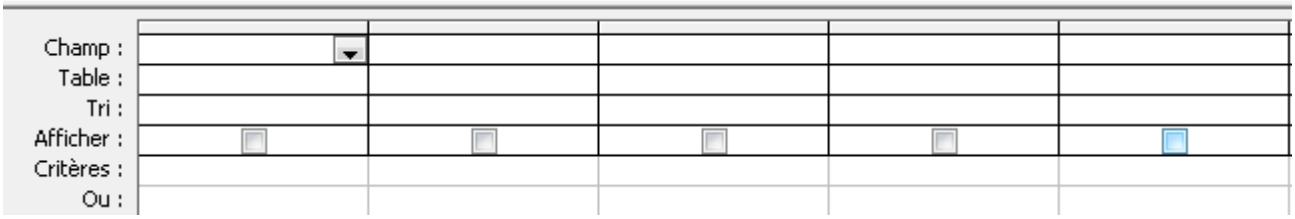
R2. ……



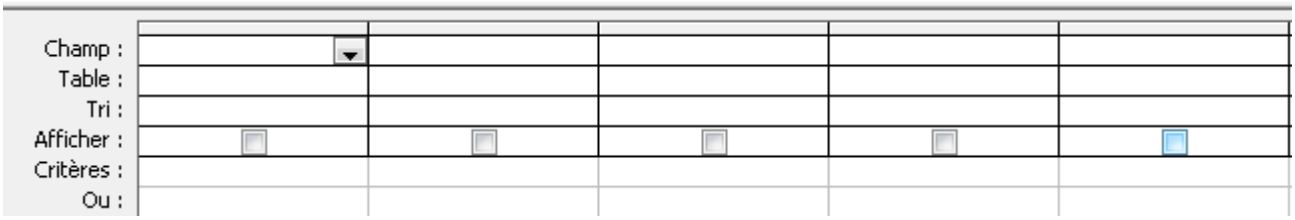
R3. ……



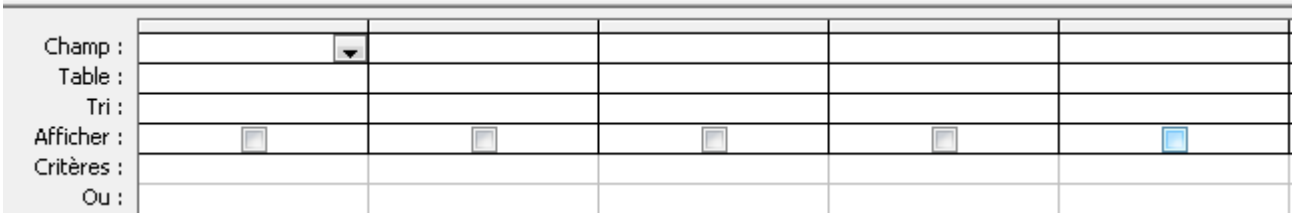
R4. ……



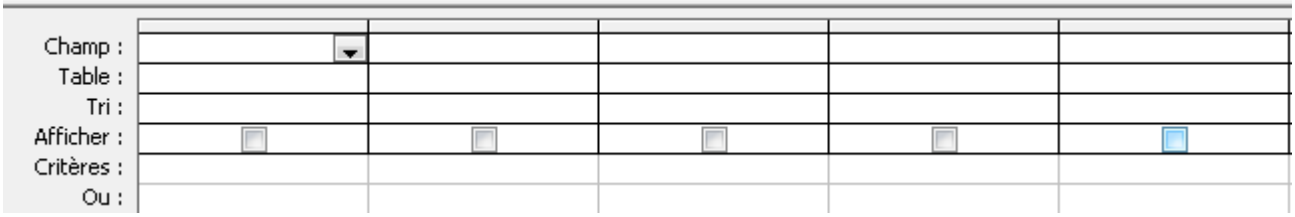
R5. ……



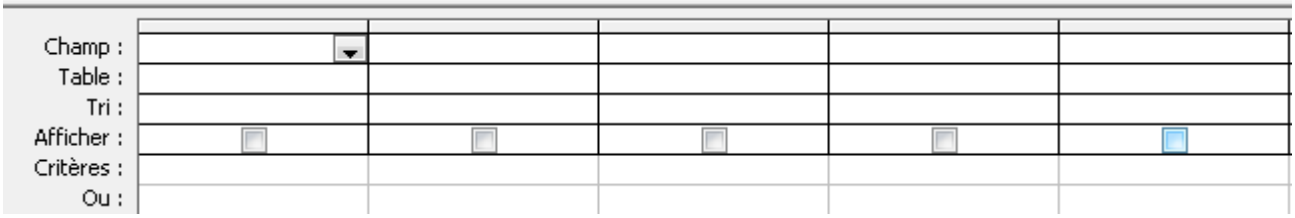
R6. ……



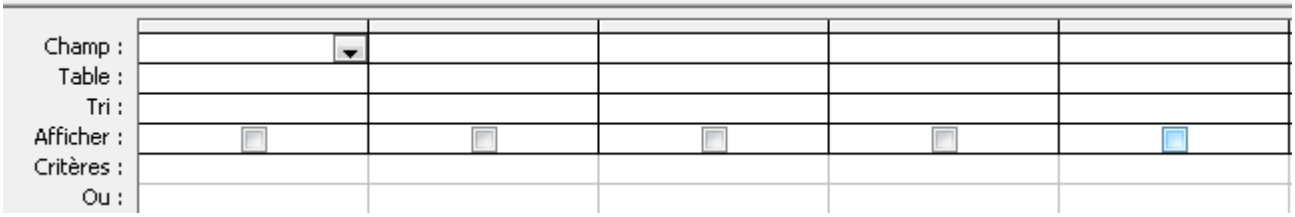
R7. ……



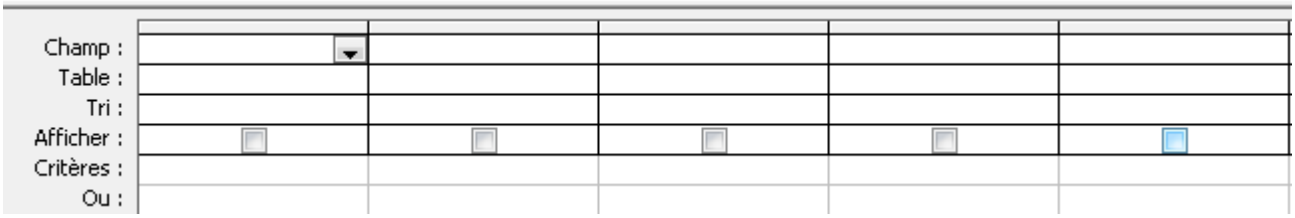
R8. ……

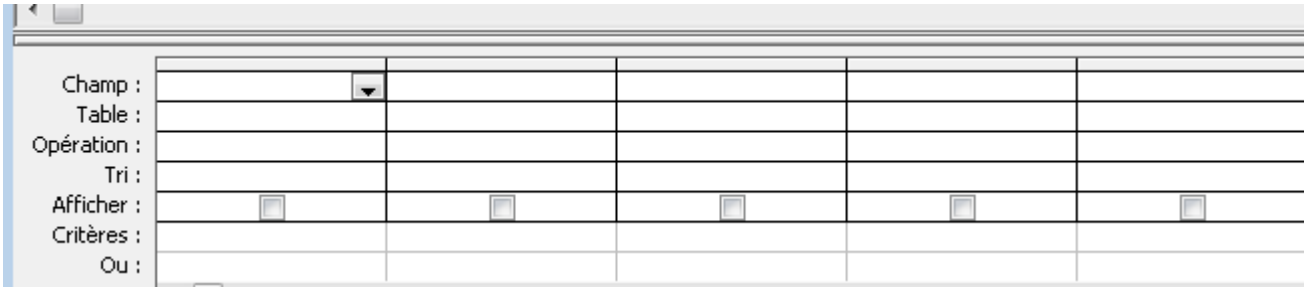


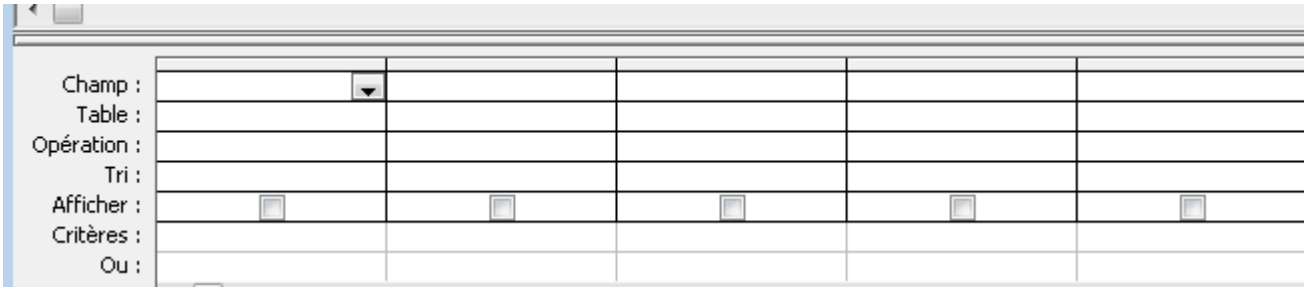
R9. ……

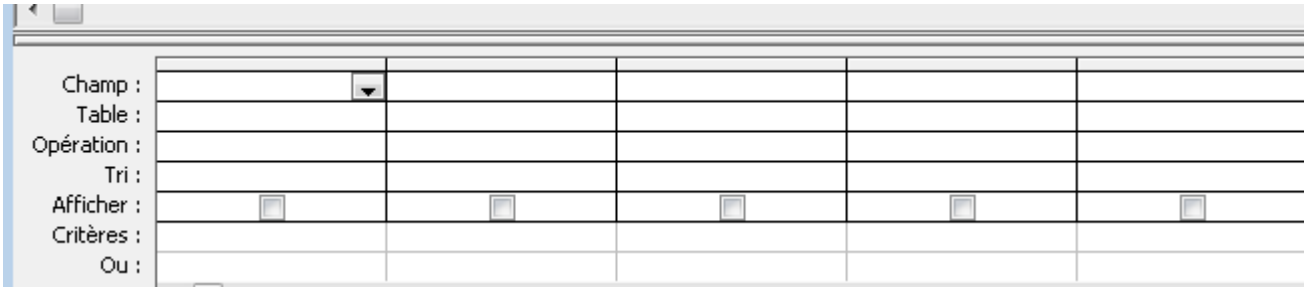


R10. ……




Champ :
Table :
Opération :
Tri :
Afficher :
Critères :
Ou :


Champ :
Table :
Opération :
Tri :
Afficher :
Critères :
Ou :


Champ :
Table :
Opération :
Tri :
Afficher :
Critères :
Ou :