Les fonctions et requêtes action

Note : les solutions sont en fin de document

1 Rappels importants

Toute requête renvoie une table comportant un ou plusieurs attribut et zéro ou plusieurs enregistrements.

Dans le cas où la requête renvoie un seul attribut et un seul enregistrement, le contenu une **valeur qui peut être utilisée telle que** dans une expression d'une autre requête.

La requête suivante ...

```
select * from facture
where facNoFou=(
      select fourNo from Fourniseur where fourNom="toto"
) ;
```

... est divisible en plusieurs sous-parties.

```
a) select * from facture where facNoFou=(QUELQUE CHOSE);
```

et

```
b) select fourNo from Fourniseur where fourNom="toto" ;
```

La partie b) donne une table résultat ...

```
select fourNo from Fourniseur where fourNom="toto" ;
```

<fourNo>	<fourNom>	suite des champs
F001	premier fournisseur	...
F003	3ème Fournisseur	...
F023	**toto**	...
F045	je suis le fournisseur 45	...
F123	1,2,Troy	...

soit la table résultat : F023 (c'est une valeur)

... qui est substituée (remplace et utilisée) dans la partie a) ...

```
select * from facture where facNoFou=("F023");
```

<facNo>	<facDate>	<facMontant>	<facMontantRegle>	<facNoFou>
00000001	19/01/05	15000 €	15000 €	**F023**
00000002	19/01/05	12500 €	12500 €	F001
00000003	20/01/05	1000 €	0 €	F023
00000004	21/01/05	22250 €	22250 €	**F123**
00000005	01/02/05	14000 €	10000 €	F001

... pour donner le résultat ci-dessous :

<facNo>	<facDate>	<facMontant>	<facMontantRegle>	<facNoFou>
00000001	19/01/05	15000 €	15000 €	F023
00000003	20/01/05	1000 €	0 €	F023

Dans le cas où une requête renvoie une seule colonne, le contenu peut être utilisé **comme une liste pour l'opérateur IN** comme suit :

```
select * from fournisseur where fourNo IN (
      select fourNo from facture where facMontant>facMontantRegle
) ;
```

La partie en jaune donne le résultat suivant :

<facNo>	<facDate>	<facMontant>	<facMontantRegle>	<facNoFou>
00000001	19/01/05	15000 €	15000 €	F023
00000002	19/01/05	12500 €	12500 €	F001
00000003	20/01/05	1000 €	0 €	F023
00000004	21/01/05	22250 €	22250 €	F123
00000005	01/02/05	14000 €	10000 €	F001

soit la table liste : F023,F001

Qui est réutilisé dans la partie bleue comme suit :

```
select * from fournisseur where fourNo IN (
      F023,F001
) ;
```

Le résultat final est :

<fourNo>	<fourNom>	suite des champs
F001	premier fournisseur	...
F023	toto	...

Par ailleurs, il existe des clauses ensemblistes qui permettent d'assembler, soustraire ou faire l'intersection de deux requêtes :

UNION : union de deux requêtes.

INTERSECT : intersection, ne sont affichées que les lignes présentes dans les deux requêtes.

MINUS : différence, le résultat sont les lignes de la première requête dont on a supprimé celles qui sont aussi dans la seconde.

Note : les résultats de chaque requête doivent avoir la même forme (mêmes type et nombre d'attributs).

Note 2 : les clauses Intersect et minus ne sont pas utilisables dans ms-Access, il faut faire autrement (cf exo sur les opérateurs ensemblistes).

2 Les expressions entre attributs

Lors de requêtes, il peut être nécessaire d'effectuer des calculs sur les attributs.

Exemple : Calcul de tva. Soit un catalogue, donnez la liste des tarifs HT, le montant de la TVA et le prix TTC des articles. Rédigez les requêtes.

a) taux fixe : TVA=19,6%

Article(codeA, descA, prixHT)

b) Le taux de TVA dépend du type de taux de l'article.

Article(codeA, descA, prixHT, typeTVA)

TauxTaxe(codeT, tauxTaxe)

Il est possible d'utiliser tous les opérateurs mathématiques courants (+, -, *, /) entre des attribut et des valeurs ou d'autres attributs.

3 Les fonctions d'agrégat d'enregistrements en SQL

Dans certains cas, nous désirons faire apparaître des totaux ou autres calculs sur des valeurs. Pour cela, on utilise des fonctions.

Ces fonctions sont à utiliser conjointement à la clause **GROUP BY**

Certaines fonctions ne sont utilisables que sur des attributs numériques (sinon résultats inattendus ou erreur).

3.1 Le comptage : count(*)

Il sert à compter le nombre d'enregistrements.

Exemple : rédigez la requête permettant de compter le nombre d'articles ;

Article(codeA, descA, prixHT, …)

3.2 Les fonctions courantes : avg(a), sum(a)

Elles servent à calculer, respectivement, la moyenne et la somme d'un attribut d'une table

Exemple : calculer la somme des factures du fournisseur 10 ainsi que le montant moyen des factures ;

Facture(facNo, facNoFou, facMontantHT, facTVA, …)

3.3 Les fonctions spécialisées : max(a), min(a)

Renvoient le(s) enregistrement(s) ayant pour valeur de l'attribut a la valeur maximum/minimum de l'ensemble des enregistrements.

Exemple : Facture(facNo, facDate, facMontantHT, facTVA, facNoFou, …)

1) donnez les numéros des dernières factures reçues.

2) donnez les numéros et montants de la première et de la dernière facture du fournisseur 10

3.4 La clause having

Cette clause est très utile lorsqu'on veut effectuer une sélection des lignes dans la table résultat de la requête.

Exemple :

Rédigez la requête qui affiche l'article "AZ345" sans utiliser la clause Where.

Donnez la liste des catégories de TVA qui comportent plus de 10 articles.

On effectue la requête de comptage et on ajoute une clause Having pour éliminer les lignes qui ne correspondent pas à la demande.

Donnez la liste des clients qui ont passé au moins une commande de plus se 5000€

Client(cliNo, …)

Clé primaire : cliNo

Commande(cdNo, … , cliNo)

Clé primaire : cdeNo

Clé étrangère : cliNo référence cliNo dans Client

LigneCdeArticle(cdeNo, artCode, cdePVenteHT, cdeQté)

Clé primaire : cdeNo, artCode

Clé étrangère : cdeNo référence cdeNo dans Commande

Clé étrangère : artCode référence artCode dans Article

La clause **having** est généralement associée à la clause **group by** qui est nécessaire pour les agrégats.

4 Les requêtes action ; delete, update, insert

Les requêtes action sont des requêtes qui modifient le contenu de la base de données.

Elles concernent :

- la suppression d'enregistrements,
- la modification d'enregistrements.
- l'insertion de nouveaux enregistrements dans une table,

Les clauses respectives sont :

- **delete from** table where … ;
  -> **Attention**, pas de champs derrière delete !

- **update** table **set** attribut1=nouvelleValeur, attribut2=expression, … where … ;
  -> **Attention**, pas de from derrière update !

- **insert into** table (Attribut1, Attribut2, Attribut3, …) **values**(valeurAttribut1, valeurAttribut2, valeurAttribut3, …) ;
- **insert into** table [(Attribut1, Attribut2, Attribut3, …)] **select ... ;**

Dans la requête insert, il est possible d'indiquer la liste des attributs à remplir si on ne dispose pas de toutes les valeurs. Attention à renseigner complètement la clé primaire.

Exemples : rédiger les requêtes suivantes (Article(codeA, descA, prixHT, …))

- Supprimer les articles obsolètes qui contiennent le mot "Solex" dans la désignation.
- L'article "roue 13X153 H" a été valorisé à zéro par erreur. Son prix doit être de 30€
- Ajouter le nouveau "voyant avant halogène" VAH001Arg au prix de 14,34€ dans Article.
- Ajouter les commandes (n° de commande et total HT) dans la table HistoCde(cdeNum, cdeTotHT). On dispose des tables : Article et CdeLigne(cdeNum, codeA, qtéCdeArt)

Attention, la virgule dans les nombres est notée '.'

5 Les vues ; create/drop view

Une vue est une pseudo-table, que l'on peut manipuler comme une table, sur laquelle on ne peut pas faire de requête action (ajout, modif, supression).

Elles permettent de créer des tables dont le contenu est le résultat d'une requête.

Exemple :

```
Create view VArtFou as
select descA, fouNom
from article, Fournisseur
where …
;
```

Liste des désignations et noms des fournisseurs des articles.

Elles peuvent remplacer avantageusement des requêtes sélection dans une requête imbriquée complexe et éviter de refaire régulièrement la même requête à de multiples endroits, avec tous les risques d'erreurs d'écriture que cela comporte.

Pour supprimer une vue :

```
Drop view VArtFou ;
```

6 Les requêtes de manipulation de tables ; create, alter, drop

Comme nous pouvons manipuler des enregistrements, nous pouvons manipuler des tables.

Les clauses suivantes : create table, alter table et drop table servent à créer, modifier et détruire une table.

La syntaxe est la suivante :

```
Drop table NomDeLaTable ;
Create table NomDeLaTable ( liste_des_champs_des_clés_et_des_contraintes );
Alter table NomDeLaTable (liste_des_modif );
```

La liste des champs contient pour chaque champ :

Nom_du_champ type longueur contraintes

Les lignes sont terminées par une virgule (sauf la dernière)

Les contraintes expriment si un champ (unique) est clé primaire ou/et clé étrangère, si la valeur NULL est autorisée ou non.

La liste des clés est utilisée si la clé primaire est composite.

Des contraintes peuvent être appliquées à la table.

La liste des modifications rappelle le champ et correspond aux nouvelles valeurs de type, longueur et contraintes à appliquer.

7 Les droits sur les bases de données (en cours de conception)

Pour limiter l'accès aux données, il est possible de mettre en place des droits.

Ces droits sont attribués à différents types d'utilisateurs, en fonction de leur rôle.

7.1 Les acteurs

L'administrateur possède les droits sur la base et les utilisateurs mais doit se faire accorder les droits sur les tables par leur(s) créateur(s).

Les concepteurs peuvent avoirs et transmettre leur droits sur les tables aux autres acteurs.

Les utilisateurs on des droits d'accès aux données plus ou moins importants (création, interrogation, modification ou suppression – **CIMS**), et plus ou moins étendus selon leurs attributions.

Ces droits sont accordés ou refusés lors de la rédaction du modèle logique de données.

7.2 La structure

Les droits peuvent des structurer en groupes.

On peut accorder des droits à un acteur particulier ou à un groupe auquel est rattaché un ou plusieurs acteurs.

7.3 Accorder ou supprimer des droits ; grant, revoke

La clause pour accorder des droit de modification de contenu sur une table à un acteur "toto" est la suivante :

```
Grant toto on maTable select, insert, update, delete ;
```

Toto aura tous les droits sur le contenu de la table maTable.

Pour supprimer des droits, on utilise la clause revoke avec la même syntaxe.

Annexe 1 Solutions de quelques exercices ou exemples

1

```
Select codeA, descA, prixHT*(1+0,196) from Article ;
```

```
Select codeA, descA, priHT*(1+tauxTaxe/100) from Article, TauxTaxe
Where typeTVA=codeT ;
```

2

```
Select count(*) from Article ;

Select  sum(facMontantHT*facTVA), avgfacMontantHT*facTVA)  from  Facture
;
```

3

```
Select facNo, max(facDate) from Facture ;

Select facNo facMontantHT from Facture
Where facNo in  (Select max(facNo) from Facture where facNoFou=10
                 union
                 Select min(facNo) from Facture where facNoFou=10
                ) ;
```

4

```
delete from Article where descA like "%Solex%" ;

update Article set prixHT=30 where descA=roue13X153 H" ;

insert into Article (codeA, descA, prixHT) Values("VAH001Arg", "voyant
avant halogène", 14.34) ;

insert into HistoCde (cdeNum, cdeTotHT)
       select cdeNum, sum(qteCdeArt*prixHT)
       from CdeLigne, Article
       where CdeLigne.codeA=Article.codeA
       group by cdeNum ;
```

ou

```
insert into HistoCde
       select cdeNum, sum(qteCdeArt*prixHT) as cdeTotHT
       from CdeLigne, Article
       where CdeLigne.codeA=Article.codeA
       group by cdeNum ;
```