

A la découverte d'Access 2007 : Les champs multi-valués



par Christophe WARIN

Date de publication : 12/05/2006

Dernière mise à jour :

Découvrez et jugez une nouveauté importante de la nouvelle version annoncée de Microsoft Access V12 (Access 2007).

Cet article est aussi bien destiné au développeur débutant souhaitant se familiariser avec les possibilités d'Access qu'au développeur professionnel désirant se faire son propre opinion sur le nouveau moteur de base de données Access.

- I - Introduction
- II - Les champs multi-valués : une révolution ?
- III - Création d'un champ multi-valué par l'exemple
- IV - Représentation des données et interrogation SQL
 - IV-A - Représentation hiérarchique
 - IV-B - Requêtes SQL : Sélection
 - IV-C - Requêtes SQL : Action
- V - Programmer les champs multi-valués
- VI - Performances
- VII - Conclusion

I - Introduction

Si depuis longtemps nous étions habitués à travailler avec le moteur Jet - en effet les dernières versions d'Access n'ont pas engendré de modifications significatives - il faut se rendre à l'évidence : Access n'est plus le même. Et pour cause, **son moteur n'est plus Jet**. L'apparition d'un nouveau moteur engendre donc un flot de nouveautés. Parmi elles : l'apparition de ce que l'on appelle les **champs multi-valués** destinés à simplifier le traitement des relations plusieurs-à-plusieurs.

Mais ce qui semble apparaître comme une réelle révolution est il à la hauteur de ce que l'on est en droit d'attendre ou bien n'est-ce qu'un simple gadget ?

II - Les champs multi-valués : une révolution ?

Oui, il est possible de parler de révolution. En effet, si depuis des années Access et son moteur de base de données sont restés compatibles avec les méthodologies de conception (notamment *Merise*), Access 12 permet des fantaisies qui feront fuir les puristes.

Vous trouvez peut-être ce discours péjoratif, mais il n'en est rien. Que les fans de *Merise* partent en courant s'ils veulent, toujours est il que cette notion de champs multi-valués est destinée à grandement simplifier le travail des développeurs que nous sommes. Remarquez par vous même à l'aide d'un exemple. Prenons le cas du système d'information d'une école. Le directeur souhaite un outil permettant d'historiser l'ensemble des classes fréquentées par un élève.

Avant

Chaque occurrence du système d'information devait être traité en ligne. Ainsi, nous disposions alors d'une table *Eleve*, d'une table *Classe* ainsi que d'une table de jointure *EleveClasse* (table que nous aurions pu nommer " *Composer* "). L'utilisation de la table de jointure amenait des requêtes souvent complexes où l'imbrication des **JOIN** devenait cauchemardesque.

Aujourd'hui

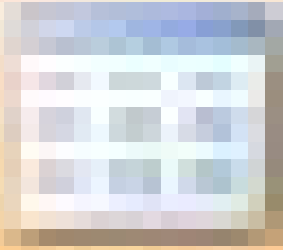
Le cauchemar pourrait bien se transformer en rêve exhaussé. Le nouveau moteur nous permet d'abandonner la table de jointures et d'obtenir la structure suivante :

Une table *Eleve* et une table *Classe* dont un champ (champ multi-valué) regroupera la liste des élèves de la classe. Visuellement, notre champ affichera la liste des identifiants des élèves séparés par des points-virgules.

Certes vous allez me dire que cette façon de faire aurait été possible avec les anciennes versions. Oui, à condition de stocker la liste des élèves dans un champ de type texte pour ainsi avoir : "1;3;5". Cependant quelle requête auriez vous écrite pour obtenir la classe de l'élève 3 ? Il aurait fallu faire appel à des fonctions VBA d'extraction de chaînes de caractères. Bref, rien de bien performant.

III - Création d'un champ multi-valué par l'exemple

Tout d'abord, créons notre table tbl_Eleve :

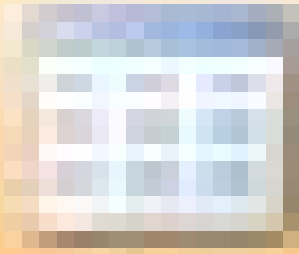


tbl_eleve :

Data Type	
Number	

Jusque là rien de nouveau, il s'agit de la création simple d'une table.

Créons maintenant la table `tbl_classe` :



tbl_cla

ata Type

umber

er

er

Le champ Eleves est logiquement de type numérique. En effet, il représente l'identifiant d'un élève qui lui est numérique (NumAuto).

Vous noterez que j'ai volontairement appliqué le pluriel au nom de mon champ. En effet, cette convention permettra de repérer facilement qu'il s'agit d'un champ qui contient plusieurs valeurs (plusieurs élèves)

Mais à ce stade, le champ n'est pas encore multi-valué. Il s'agit pour l'instant d'un champ numérique standard stockant un entier long.

Pour définir ainsi ce champ, il faut accéder à l'onglet **Lookup** (Recherche dans la version française) de la page de propriété du champ. Comme dans les versions actuelles, il est alors possible de définir quel contrôle permettra l'affichage du champ dans la table. Je choisis une zone de liste (*Display Control = ListBox*). Je renseigne la requête source de la zone de liste

```
SELECT ID, Nom, prenom FROM tbl_eleve ORDER BY Nom;
```

Cette requête établit la liste des élèves présents dans la table tbl_Eleve.

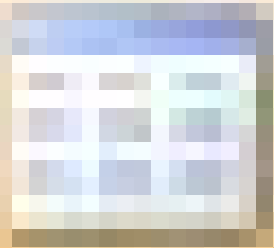
Enfin, il est nécessaire de fixer la propriété **Allow Multiple Value** (*Plusieurs valeurs autorisées*) à **Yes** (*Oui*).

Le champ admet alors plusieurs valeurs qu'il affichera dans une zone de liste.

ery

0, Nom, prenom FR

Sauvegardons la table, créons un jeu d'essai dans la table tbl_Eleve et visualisons la table tbl_classe .



tbl_class

El

Nom

NFRRFY

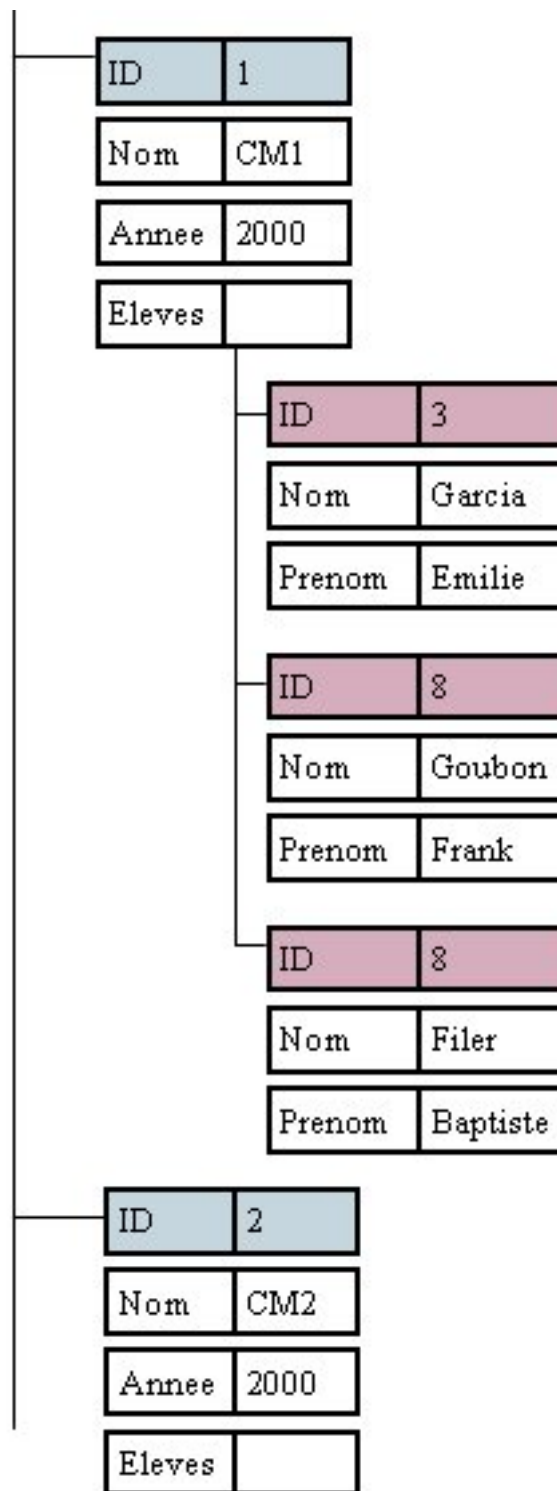
Comme vous pouvez le voir, la saisie des différents élèves de la classe est vraiment très simple, il suffit de cocher la case correspondante.

IV - Représentation des données et interrogation SQL

IV-A - Représentation hiérarchique

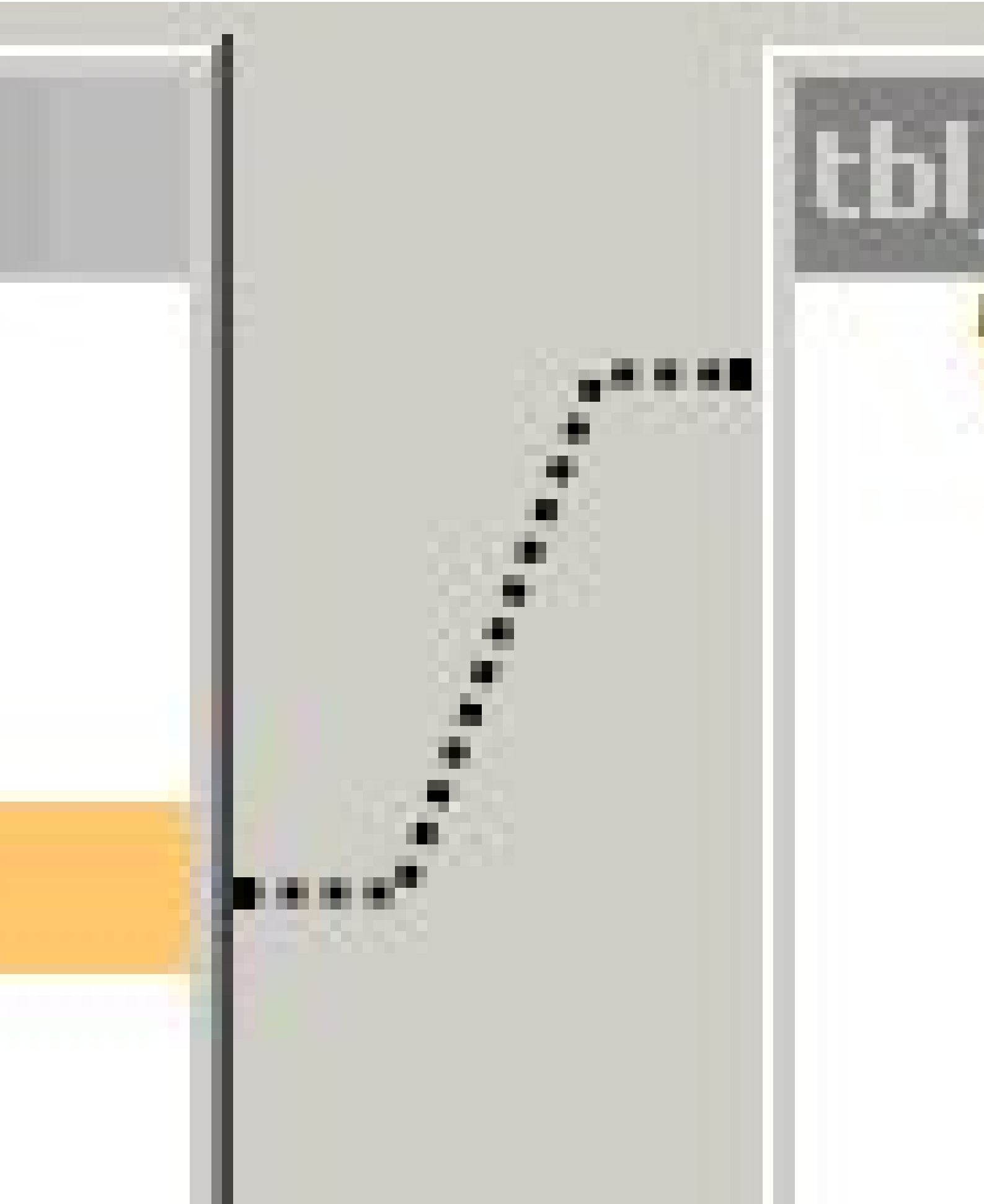
Si la mise en place d'un tel système est simple, il est impératif de comprendre comment le moteur de base de données accède aux informations.

Il faut en fait s'imaginer le champ Eleves comme une sous-table. Il pourrait ainsi être comparé à :



On obtient donc, ici, une structure hiérarchique des données. Le premier niveau constitue la table `tbl_classe`. Les données sont donc accessibles par le nom de la table. Jusqu'ici, rien de nouveau par rapport aux autres versions d'Access. En revanche, nous bloquons pour obtenir une ligne du champ `Eleves`. C'est à dire une ligne du deuxième niveau. Pour cela les développeurs du moteur de base de données ont implémenté une nouvelle propriété permettant de travailler sur les données d'un champ multi-valué. Il s'agit de la propriété **Value** du champ.

Ainsi, la relation mise en place entre les tables de notre schéma est :



Il existe donc une différence fondamentale entre **Eleves** et **Eleves.Value**.

Le meilleur moyen de l'illustrer est d'utiliser deux requêtes :

```
SELECT Eleves  
FROM tbl_classe
```

Eleves

```
SELECT Eleves.Value  
FROM tbl_classe
```

Elevés. Valus

Ainsi, la première requête retourne un champ multi-valué, la seconde retourne chacune des valeurs du champ multi-valué. En d'autres termes, la propriété **Elevés.Value** retourne un jeu d'enregistrements.

IV-B - Requêtes SQL : Sélection

Une fois cette distinction de la propriété **Value** faite, il sera naturellement possible de travailler sur les données avec des requêtes SQL.

Ainsi, la requête retournant la liste des classes fréquentées par l'élève possédant l'identifiant 2 sera :

```
SELECT *  
FROM tbl_classe  
WHERE Eleves.Value=2
```



Elev

Celle listant les classes fréquentées par l'élève 2 mais pas par le 12 sera :

```
SELECT *  
FROM tbl_classe  
WHERE Eleves.Value=2 AND Not Eleves.Value=12
```



Enfin, l'affichage complet des élèves et des classes sera obtenu par :

```
SELECT tbl_classe.NomClasse, tbl_eleve.Nom, tbl_eleve.Prenom, tbl_classe.Annee
FROM tbl_classe
INNER JOIN tbl_eleve
ON tbl_classe.Eleves.Value = tbl_eleve.ID;
```

AIN	Pre
IN	Paul
INT	Lucie
INT	Lucie
IA	Emilie
ON	Tom

IV-C - Requêtes SQL : Action

Là, nous rencontrons un problème majeur. En effet, il ne faut pas oublier que le champ multi-valué représente une association *plusieurs-à-plusieurs*. Dès lors, difficile d'envisager des requêtes **Insert** et **Update** sans pouvoir vérifier au préalable que l'enregistrement correspondant est bien présent dans la table attachée par la relation.

Seule une requête **Delete** peut encore être intéressante :

Exemple de suppression de l'élève n°2 de la classe CP :

```
DELETE Eleves.Value FROM tbl_classe  
WHERE Eleves.Value=2 AND NomClasse="CP"
```

V - Programmer les champs multi-valués

Comme vous le savez, un des atouts d'Access est le langage VBA qui l'accompagne. En outre les objets du modèle **DAO (Data Access Objects)** permettent un contrôle fin des données et de la structure de la base. Il est évident que Microsoft a pensé à rendre ce modèle compatible avec la nouvelle notion de champ multi-valué.

Plus haut nous avons introduit la propriété **Value** d'un champ multi-valué comme étant un jeu d'enregistrements. Naturellement, cette propriété se traduit par un objet **DAO.Recordset** en VBA.

Ainsi, il est possible d'avoir :

```
Dim oRst as DAO.Recordset
Set oRst=oRstClasse.Fields("Eleves").Value
```


Où **oRstClasse** correspond à un **Recordset** ouvert sur la table **tbl_Classe**.

Si vous avez trouvé les requêtes concernant les champs multi-valués un peu complexes, il va de soit que cette représentation logique sous forme de Recordset va grandement faciliter les manipulations des champs multi-valués. Il sera ainsi possible de manipuler le recordset issu du champ comme n'importe quel autre Recordset.

Un exemple pour illustrer mes propos :

Ajoutons l'élève n° 2 à la classe CP de 2000

```
Dim oDb As DAO.Database
Dim oRstClasse As DAO.Recordset, oRst As DAO.Recordset
'Instancie l'objet Database
Set oDb = CurrentDb
'Instancie un recordset correspondant à la classe CP de 2000
Set oRstClasse = oDb.OpenRecordset( _
    "SELECT * FROM tbl_Classe WHERE NomClasse='CP' AND Annee=2000")
With oRstClasse
    'Vérifie que le curseur n'est pas vide
    If Not .EOF Then
        'Récupère le recordset du champ multi-valué
        Set oRst = .Fields("Eleves").Value
        'Ajoute l'élève 2
        With oRst
            .AddNew
            .Fields(0) = 2
            .Update
        End With
    Else
        MsgBox "Aucune classe valide", vbCritical, "Erreur"
    End If
End With
```

Comme vous pouvez le constater, il s'agit d'opérations simples appliquées aux objets DAO. Aussi, pour plus de détails je vous renvoie vers ce cours :  **Définition et manipulation des données en VBA avec DAO**.

Il est à noter toutefois que cette nouvelle façon de représenter une association facilite l'insertion d'enregistrements dans la table principale ainsi que dans l'association.

Avant, la création d'une nouvelle classe aurait suivi le schéma suivant :

- 1 Création de la nouvelle classe
- 2 Valorisation des champs de la nouvelle classe
- 3 Récupération de l'identifiant de la classe
- 4 Validation de l'enregistrement dans la table classe
- 5 Création des enregistrements fils en tenant compte de la valeur de la clé étrangère (identifiant de la classe précédemment créée)

Aujourd'hui, la relation entre les classes et les élèves est représentée par un unique champ dans la table classe. De ce fait, il sera possible d'affecter les élèves à la classe pendant la création de cette dernière :

```
Dim oDb As DAO.Database
Dim oRstClasse As DAO.Recordset, oRst As DAO.Recordset
'Instancie l'objet Database
Set oDb = CurrentDb
'Instancie un recordset correspondant à la table tbl_classe
Set oRstClasse = oDb.OpenRecordset("tbl_classe")
With oRstClasse
    'Ajoute une nouvelle classe
    .AddNew
    .Fields("NomClasse") = "CE2"
    .Fields("Annee") = 2002
    'Récupère le recordset du champ multi-valué
    Set oRst = .Fields("Eleves").Value
    'Ajoute l'élève 2
    With oRst
        .AddNew
        .Fields(0) = 2
        .Update
    End With
    .Update
End With
```


Là, où cela devient vraiment très intéressant, c'est si, finalement, vous désirez annuler la mise à jour de l'enregistrement. Dans ce cas, un simple **CancelUpdate** sur le recordset de la table *tbl_classe* provoquera automatiquement l'abandon des données de l'association liant *tbl_classe* et *tbl_eleve*.

```
Dim oDb As DAO.Database
Dim oRstClasse As DAO.Recordset, oRst As DAO.Recordset
'Instancie l'objet Database
Set oDb = CurrentDb
'Instancie un recordset correspondant à la table tbl_classe
Set oRstClasse = oDb.OpenRecordset("tbl_classe")
With oRstClasse
    'Ajoute une nouvelle classe
    .AddNew
    .Fields("NomClasse") = "CM1"
    .Fields("Annee") = 2003
    'Récupère le recordset du champ multi-valué
    Set oRst = .Fields("Eleves").Value
    'Ajoute l'élève 2
    With oRst
        .AddNew
        .Fields(0) = 2
        .Update
    End With
    'Annule la mise à jour
    .CancelUpdate
End With
```

Il est ainsi possible d'éviter la programmation manuelle d'une transaction et d'un rollback.

VI - Performances

Afin de terminer cette étude, il reste à se pencher sur les performances offertes par cette nouvelle fonctionnalité.

 *Les tests ont été effectués avec la version bêta 1 d'Office 12. Il ne faut pas oublier qu'une telle version enregistre des informations de débogage destinées aux équipes de développement. La collecte de telles informations pénalise donc les performances.*

Pour cela, nous allons comparer le temps de traitement d'une requête sur ce schéma utilisant les champs multi-valués à celui d'une autre requête sur un autre schéma utilisant une relation plusieurs-à-plusieurs.

Les deux schémas sont les suivants :

lasse

IDClasse

NomClasse

Elevés

Elevés.Value

Année

La requête en question consistera à l'affichage des données (classes et élèves) correspondant à l'enfant n° 2.

La requête travaillant avec les champs multi-valué sera donc :

R02_TestMultiValue

```
SELECT tbl_classe.NomClasse, tbl_eleve.Nom, tbl_eleve.Prenom, tbl_classe.Annee
FROM tbl_classe INNER JOIN tbl_eleve ON tbl_classe.Eleves.Value = tbl_eleve.ID
WHERE tbl_eleve.ID=2
```

Celle travaillant avec la relation plusieurs-à-plusieurs sera :

R03_TestRelation

```
SELECT *
FROM tbl_Eleve2 INNER JOIN (tbl_Classe2
    INNER JOIN tbl_composer ON tbl_Classe2.ID = tbl_composer.IDClasse)
    ON tbl_Eleve2.ID = tbl_composer.IdEleve
WHERE tbl_Eleve2.ID=2
```

La procédure de test utilisée pour comparer les temps de traitement est la suivante :

```
Sub test()
Dim oDb As DAO.Database
Dim oRst As DAO.Recordset
Dim T As Double
Dim I As Integer
Set oDb = CurrentDb
T = Timer
For I = 1 To 1000
    Set oRst = oDb.QueryDefs("R02_TestMultiValue").OpenRecordset
    oRst.MoveNext
Next I
Debug.Print "R02 exécutée en : " & Timer - T & " s"
T = Timer
For I = 1 To 1000
    Set oRst = oDb.QueryDefs("R03_TestRelation").OpenRecordset
    oRst.MoveNext
Next I
Debug.Print "R03 exécutée en : " & Timer - T & " s"
End Sub
```

Il s'agit en fait d'un appel itératif des deux requêtes.

Résultats :

```
R02 exécutée en : 10,8913750000065 s
R03 exécutée en : 10,4378750000033 s
R02 exécutée en : 10,7035000000033 s
R03 exécutée en : 10,1723749999946 s
R02 exécutée en : 10,7504999999946 s
R03 exécutée en : 10,5473749999946 s
R02 exécutée en : 10,6571249999979 s
R03 exécutée en : 10,4220000000059 s
```

Comme vous pouvez le constater, il n'y a pas de grande différence sur un tel jeu d'essai. Notons toutefois que l'utilisation des champs multi-valués semble très légèrement moins performante.

Toutefois, impossible de se limiter à un tel test sur si peu d'enregistrements. Aussi, sur un jeu beaucoup plus vaste (200 classes pour 10000 élèves et en moyenne 10 élèves par classe), il s'avère que l'utilisation des champs multi-valués devient réellement pénalisante. Pour preuve, un seul appel de la requête R03 est exécuté en 0.8 seconde, alors que l'appel de R02 demande plus de 10 secondes.

VII - Conclusion

A travers cet article de présentation des champs multi-valués nous avons pu mettre en évidence la simplification du schéma de données qu'ils occasionnent. S'il est vrai que l'utilisation de SQL avec ce nouveau type de données se révèle plutôt hasardeuse, le développeur aguerri saura profiter de la puissance de DAO afin de tirer la meilleure partie de cette nouvelle approche. J'émets juste quelques réserves quant à l'utilisation par le développeur débutant. En effet il ne faudrait pas que cette simplification du modèle de données tente à masquer des erreurs de conception ou bien encore la peur d'écrire des requêtes complexes.

Toutefois, je reste assez sceptique quant à l'adoption de cette nouvelle méthode par les professionnels. En effet comme on me l'a fait remarquer à juste titre, le professionnel génère ses schémas de base de données à l'aide d'outils de modélisation (notamment **CaseStudio**). De ce fait, les fichiers résultant n'utiliseront pas cette nouveauté d'Access mais conserverons une structure plus normalisée qui saura faciliter un passage d'un SGBD à un autre.

Enfin rappelons que du fait des performances offertes par cette nouvelle méthode, l'utilisation des champs multi-valués devra être réservée aux tables contenant très peu de données à moins que la version commerciale soit beaucoup plus rapide que la bêta 1.

