- ACCES AUX DONNEES AVEC DAO -

En Visual Basic, plusieurs méthodes d'accès aux bases de données sont proposées au programmeur :

· Le contrôle "Data" : simple d'utilisation, pour les programmes les moins évolués qui accèdent aux bases de données le plus souvent locales,

· D.A.O (Data Access Object) très utilisé pour l'accès aux bases de données locales,

· A.D.O (Active X Data Object) : très utilisé pour l'accès aux bases de données distantes.

Sachez aussi à titre d'informations qu'il existe aussi les méthodes "A.D.O.X" qui offre des facilités en matière d'administration (intéressant pour les bases de données en ligne) et de sécurité, "J.R.O" offrant des commandes avancées en matière de réplication de données et R.D.O pour les bases de données des réseaux locaux. Ces trois méthodes ne seront pas traitées dans le chapitre.

Dans ce chapitre, on fait souvent appel à la notion de jeu d'enregistrements (encore appelé recordset). Un enregistrement contient l'ensemble des valeurs d'un tuple, c'est à dire qu'il contient une valeur pour chaque champs. Un jeu d'enregistrements est l'ensemble des enregistrements qui sont contenus dans une table. Il existe différents types de jeux d'enregistrements. Ils seront traités plus loin dans la sections "Type des jeux d'enregistrements", dans les chapitres sur le Data, DAO et ADO, on utilise le type par défaut "Table" pour une base de données de type Access et "Forward Only" pour une base de donnée de type ODBC puisque l'on spécifie rien lors de l'ouverture des recordsets.

Exemple :

NOM	**PRENOM**
DUPONT	Laurent
HUMETZ	Mélanie
MARTIN	Jérôme

1 enregistrement = un tuple = "DUPONT","Laurent" par exemple
Enregistrement n°2 = "HUMETZ", "Mélanie"
Un jeu d'enregistrement = tous les enregistrements = "DUPONT","Laurent" ; "HUMETZ", "Mélanie"; "MARTIN", "Jérôme".

UTILISATION DU CONTROLE DATA (DATA CONTROL)

Le contrôle "Data" est disponible dans la boîte à outil standard. Il permet de se connecter aux bases de données de types Jet ou ODBC. La première chose à faire est de déposer un contrôle Data sur la Form. Ensuite, il faut renseigner un certain nombre de propriétés :

Name : nommer le contrôle Data (ou conserver son nom par défaut).
Databasename : permet de spécifier le chemin et le nom de la base de données à ouvrir.

Recordsource : permet de spécifier le nom de la table.

Une fois ces propriétés renseignées, l'accès à la base est établie. Certaines propriétés permettent de modifier les paramètres du contrôle Data. Les principales propriétés sont :

Connect : Type de base de données à ouvrir (Access, Excel …).
DefaultType : Définit si la base de données est de type Jet ou ODBC.
RecordsetType : Type d'objet recordset (encore appelé jeux d'enregistrements).
Visible : Définit si le contrôle Data est visible ou non sur la feuille en mode exécution.

Une fois la connexion à la base de données établie, il reste à afficher les jeux d'enregistrements (ou recordsets) à l'aide de TestBox. En effet, le contrôle TextBox disposent de propriétés le permettant d'être lié à un contrôle Data et donc à une base de données. Ces propriétés sont :

DataSource : Spécifie le nom du contrôle Data permettant l'accès à la base de données.
DataField : Spécifie le nom du champs dont on affiche la valeur.

Il suffit d'utiliser autant de TextBox qu'il n'y a de champs pour pouvoir afficher la totalité des jeux d'enregistrements. Utilisez les flèches du Data Control pour faire défiler les enregistrements. Du fait que la TextBox est liée au Data, son affichage est actualisé automatiquement à chaque déplacement dans le jeu d'enregistrements.

MODELE D.A.O (DATA ACCESS OBJECT)

Pour utiliser le modèle D.A.O, il faut ajouter une référence au projet, c'est à dire ajouter la bibliothèque d'objets correspondante. Pour cela cliquez sur "Références" dans le menu contextuel "Projet". Cochez la bibliothèque d'objets "Microsoft D.A.O Object Library" puis cliquez sur "Ok" pour ajouter la bibliothèque au projet.

Exemple : On a une base de données Jet située à la racine du disque dur, elle se nomme "BD1.mdb".
Elle contient deux tables : "Personne" et "Métier"

1 – Déclaration de la base de données

La première chose à faire est de déclarer la base dans le programme.
ex : **Dim** MaBase **as DataBase** (MaBase est le nom que je donne à la base dans mon programme).

2 – Déclaration des jeux d'enregistrements ou recordsets

Il est conseillé de créer autant de jeux d'enregistrements qu'il y a de tables et requêtes dans votre base de données.

Dim PersonneRst **as Recordset** (PersonneRst est le nom que je donne à mon recordset dans mon programme).

Dim MétierRst **as Recordset** (MétierRst est le nom que je donne à mon recordset dans mon programme).

3 – Ouverture de la base de données

Le plus souvent, on ouvre la base dans la procédure événementielle "Form_Load".

Set MaBase = **OpenDatabase ("**C:\BD1.mdb**")**

MaBase est le nom avec lequel j'ai déclaré ma base de données.

4 - Affectation des tables aux recordsets

Set PersonneRst = MaBase.**OpenRecordset** ("Personne")
Set MétierRst = MaBase.**OpenRecordset** ("Métier")
PersonneRst et MétierRst sont les noms avec lesquels j'ai déclaré mes recordsets

Il existe différents types de jeux d'enregistrements, ils sont décris plus bas dans la section "type d'enregistrements". Ici on ne précise rien, c'est donc le type par défaut qui est utilisé "Table" pour une base de données de type Access et
"Forward Only" pour une base de donnée de type ODBC.

5 - Navigation dans le jeu d'enregistrements

Comme on n'utilise plus de contrôle Data, les boutons de navigations dans les jeux d'enregistrements sont à créer par les développeurs. Pour cela, on va créer des boutons de commandes dans lesquels on va placer les instructions suivantes (RecordsetRst est le nom du jeu d'enregistrement) :

RecordsetRst.MoveNext	Aller à l'enregistrement suivant
RecordRst.MovePrevious	Aller à l'enregistrement précédent
RecordRst.MoveFirst	Aller au premier enregistrement
Recordset.MoveLast	Aller au dernier enregistrement
Recordset.Move + n	Déplacer de n enregistrements (n est un entier positif ou négatif)

Exemple : on crée un bouton appelé CmdSuivant dans lequel on va placer l'instruction "PersonneRst.MoveNext" sur l'événement clique, puis un bouton CmdPrécédent …

Les propriétés BOF (Begin Of File) et EOF (End Of File) respectivement début de fichiers et fin de fichiers, retournent vrai si on a atteint le premier ou le dernier enregistrement (selon le cas).

Exemple :

```
If PersonneRst.EOF = True then        'Emission d'un bip sonore si on atteint le dernier enregistrement
```

```
    Beep
End if
```

6 - Affichage des enregistrements

Comme pour l'affichage des recordsets avec le DataControl, avec D.A.O on va également utiliser les TextBox. La différence réside dans le fait que l'on ne va pas utiliser les propriétés de la TextBox pour la lier à la base. Par conséquent les TextBox ne seront pas actualisées lorsque l'on va se déplacer dans le jeu d'enregistrements. C'est au développeur de le programmer.
Exemple : on crée une TextBox nommée "TxtNom" qui va recevoir le champs "nom" de la table "personne". A chaque fois que l'on va se déplacer dans le jeu d'enregistrements, il faudra actualiser l'affichage de la TextBox. On place donc le code suivant derrière le bouton CmdSuivant :

```
PersonneRst.MoveNext                    'Suivant
TxtNom = PersonneRst("NOM")             'Actualiser
```

On peut également actualiser l'affichage avec la syntaxe suivante :

```
TxtNom = PersonneRst![NOM]
```

7 - Ajouter un enregistrement à la fin du jeu

```
PersonneRst.AddNew                      'Ajouter un enregistrement
PersonneRst![NOM] = "MARTIN"            'Ajout d'un nom
PersonneRst.Update                      'Valider l'ajout
```

Tant que la méthode "update" n'est pas lancée, l'enregistrement n'est pas ajouté.

8 - Modifier un jeu d'enregistrements

D'abord, on se place sur l'enregistrement que l'on veut modifier à l'aide des méthodes de la famille "move", puis on place les instructions suivantes :

```
PersonneRst.Edit                        'Ouverture de l'enregistrement pour modification
PersonneRst![NOM] = "MARTINE"           'Modification, ici du champs nom
PersonneRst.Update                      'Mise à jour de l'enregistrement
```

9 - Annulation d'un ajout ou d'une modification

Après l'utilisation de la méthode "Addnew" ou "Edit", on peut annuler l'opération par la méthode "CancelUpdate" :

```
PersonneRst.AddNew                      'Ajouter
PersonneRst.Cancelupdate                'Annuler
```

10 - Suppression d'un enregistrement

Un enregistrement peut être supprimé à l'aide de la méthode "Delete" :
PersonneRst.Delete 'Effacer

11 – Compter le nombre d'enregistrements dans un jeu d'enregistrements

La propriété "RecordCount" permet de comptabiliser le nombre d'enregistrements contenus dans un jeu.
Exemple : PersonneRst.Recordcount
'Retourne le nombre d'enregistrements contenus dans le recordset "PersonneRst"

12 - Fermer les jeux d'enregistrements

A l'aide de la méthode "Close" :
PersonneRst.Close 'Fermer
MétierRst.Close

13 - Fermeture de la Base de données

Egalement à l'aide de la méthode "Close" :
MaBase.close 'Fermer

14 – Utilisation de requêtes

On ouvre une requête de la même manière qu'une table, par la ligne de codes suivante :
MaBase est le nom de déclaration de la base de données.
MétierRst est le nom du recordset.

Exemple de requête non paramétrée : **Set** MétierRst = MaBase.**OpenRecordset** ("SELECT * FROM Métier")

Exemple de requête paramétrée : **Set** MétierRst = MaBase.**OpenRecordset** ("SELECT DESIGNATION FROM METIER WHERE CODE = '" & TxtCode & "'")

Remarques : le paramètre est fournit par la TextBox "TxtCode". Le code doit être obligatoirement encadré par des apostrophes (') puisque c'est un paramètre numérique. On doit donc utiliser l'opérateur de concaténation (&) pour concaténer la chaîne SQL et la valeur du paramètre en une expression unique.

MODELE A.D.O (ACTIVE X DATA OBJECT)

Pour utiliser le modèle A.D.O, il faut également ajouter une référence au projet, c'est à dire ajouter la bibliothèque d'objets correspondante. Pour cela cliquez sur "Références" dans le menu contextuel "Projet". Cochez la bibliothèque d'objets "Microsoft ActiveX Data Objects Library 2.1" puis cliquez sur "Ok" pour ajouter la bibliothèque.

Exemple : On a une base de données Jet située à la racine du disque dur, elle se nomme "BD1.mdb".
Elle contient deux tables : "Personne" et "Métier"

1 – Déclaration de la connexion

Avec A.D.O, on déclare une connexion à une base de données et non la base de données elle même.
La première chose à faire est de déclarer la connexion à la base dans le programme.
ex : **Dim** MaConnection **as new ADODB.Connection** (Maconnection est le nom que je donne à la connexion).

2 – Déclaration des jeux d'enregistrements ou recordsets

Il est conseillé de créer autant de jeux d'enregistrement qu'il y a de tables et requêtes dans votre base de données.
Dim PersonneRst As New ADODB.Recordset
(PersonneRst est le nom que je donne à mon recordset dans mon programme).
Dim MétierRst As New ADODB.Recordset
(MétierRst est le nom que je donne à mon recordset dans mon programme).

3 – Définition du Provider

Avec A.D.O, on peut ouvrir tous type de base à condition d'en détenir le Provider. Celui de Access est : **"Microsoft.jet.OLEDB.4.0"**.
L'instruction :
MaConnection**.Provider** = "Microsoft.jet.OLEDB.4.0"
Permet de définir le provider pour Access (MaConnection étant le nom de la connexion).

4 – Ouverture de la connexion

Le plus souvent, on ouvre la connexion dans la procédure événementielle "Form_Load".
MaConnection**.Open ("data source** = C:\bd1.mdb**")**
MaConnection est le nom avec lequel j'ai déclaré ma connexion à la base de données.

Avec ADO, il aussi possible d'accéder à des bases de données dont l'accès est autorisé par mot de passe. Dans ce cas, il faut spécifier le nom utilisateur (Username) et son mot de passe (Password) lors de l'ouverture de la base de données.
Exemple : MaConnection**.Open (source, username, password)**
MaConnection**.Open** ("data source=c:\bd2.mdb","Laurent","LD1980")

5 - Affectation des tables aux recordsets

Recordset**.Open** Nom du champs, NomConnection, Constante d'ouverture
Exemple : MétierRst.Open "METIER", MaConnection, adOpenDynamic

Les constantes exprimantes les mode d'ouvertures sont détaillées dans la section "type des recordsets"

6 - Navigation dans le jeu d'enregistrements

Comme on n'utilise plus de contrôle Data, les boutons de navigations dans les jeux d'enregistrements sont à créer par les développeurs. Pour cela, on va créer des boutons de commandes dans lesquels on va placer les instructions suivantes (RecordsetRst est le nom du jeu d'enregistrement) :

RecordsetRst.MoveNext	Aller à l'enregistrement suivant
RecordRst.MovePrevious	Aller à l'enregistrement précédent
RecordRst.MoveFirst	Aller au premier enregistrement
Recordset.MoveLast	Aller au dernier enregistrement
Recordset.Move + n	Déplacer de n enregistrements (n est un entier positif ou négatif)

Exemple : on crée un bouton appelé CmdSuivant dans lequel on va placer l'instruction "PersonneRst.MoveNext" sur l'événement clique, puis un bouton CmdPrécédent …

Les propriétés BOF (Begin Of File) et EOF (End Of File) respectivement début de fichiers et fin de fichiers, retournent vrai si on a atteint le premier ou le dernier enregistrement (selon le cas).
Exemple :

```
If PersonneRst.EOF = True then      'Emission d'un bip sonore si on atteint le dernier enregistrement
                Beep
End if
```

7 - Affichage des enregistrements

Comme pour l'affichage des recordsets avec le DataControl, avec A.D.O on va également utiliser les TextBox. La différence réside dans le fait que l'on ne va pas utiliser les propriétés de la TextBox pour la lier à la base. Par conséquent les TextBox ne seront pas actualisés lorsque l'on va se déplacer dans le jeu d'enregistrements. C'est au développeur de le programmer.
Exemple : on crée une TextBox nommée "TxtNom" qui va recevoir le champs "nom" de la table "personne". A chaque fois que l'on va se déplacer dans le jeu d'enregistrements, il faudra actualiser l'affichage de la TextBox. On place donc le code suivant derrière le bouton CmdSuivant.

```
PersonneRst.MoveNext                'Suivant
```

```
TxtNom = PersonneRst("NOM")          'Actualiser
```

On peut également actualiser l'affichage avec la syntaxe suivante :
TxtNom = PersonneRst![NOM]

8 - Ajouter un enregistrement à la fin du jeu

```
PersonneRst.AddNew                   'Ajouter un enregistrement
PersonneRst![NOM] = "MARTIN"         'Ajout d'un nom
PersonneRst.Update                   'Valider l'ajout
```
Tant que la méthode "update" n'est pas lancée, l'enregistrement n'est pas ajouté.

9 - Annulation d'un ajout

Après l'utilisation de la méthode "Addnew", on peut annuler l'opération par la méthode "CancelUpdate" :

```
PersonneRst.AddNew                   'Ajout
PersonneRst.Cancelupdate             'Annuler
```

10 - Suppression d'un enregistrement

Un enregistrement peut être supprimé à l'aide de la méthode "Delete" :

```
PersonneRst.Delete                   'Effacer
```

11 – Compter le nombre d'enregistrements dans un jeu d'enregistrements

La propriété "RecordCount" permet de comptabiliser le nombre d'enregistrements contenus dans un jeu.
Exemple : PersonneRst.Recordcount
'Retourne le nombre d'enregistrements contenus dans le recordset "PersonneRst"

12 - Fermer les jeux d'enregistrements

A l'aide de la méthode "Close" :

```
PersonneRst.Close                    'Fermer
MétierRst.Close
```

13 - Fermeture de la Base de données

Egalement à l'aide de la méthode "Close" :

```
MaBase.close                         'Fermer
```

14 – Utilisation de requêtes

On ouvre une requête de la même manière qu'une table, par la ligne de codes suivante :
Recordset**.Open** "Instruction SQL", NomConnection, Constante d'ouverture
Les constantes exprimantes le mode d'ouvertures sont détaillées dans la section "type des recordsets"

Exemple d'une requête non paramétrée : PersonneRst**.Open** "SELECT * FROM NOM", MaConnection, OpenForwardOnly

Exemple de requête paramétrée : MétierRst.**Open** ("SELECT DESIGNATION FROM METIER WHERE CODE = '" & TxtCode & "'", MaConnection, OpenForwardOnly)

Remarques : le paramètre est fournit par la TextBox "TxtCode". Le code doit être obligatoirement encadré par des apostrophes (') puisque c'est un paramètre numérique. On doit donc utiliser l'opérateur de concaténation (&) pour concaténer la chaîne SQL et la valeur du paramètre en une expression unique.

TYPES DES JEUX D'ENREGISTREMENTS (OU RECORDSETS)

Il existe différents types de jeux d'enregistrements en fonction de ce que l'on souhaite faire des données qu'il contient. Chaque type dispose d'une particularité intéressante permettant de favoriser l'aspect sécurité, l'aspect rapidité ou alors un compromis entre les deux. Les différents types sont :

- **Table :** Ouvre un jeu d'enregistrements de type table permettant de modifier les enregistrements d'une table de base de données de type Jet. En effet, lorsqu'on modifie le recordset, la table est donc aussi modifiée. Mode d'ouverture rapide. Avec ce type de recordset, on ne peut pas ouvrir de requêtes utilisant plusieurs tables.

- **Dynaset :** Ouvre un jeu d'enregistrements de type feuille de réponse dynamique dont les enregistrements peuvent provenir de plusieurs tables (grâce à l'utilisation de requêtes). Chaque enregistrement peut être modifié. Mode d'ouverture permettant une bonne souplesse d'utilisation. C'est le plus puissant type de recordset.

- **Snapshot (ou Instantané) :** Ouvre un jeu d'enregistrements de type statique, donc qui est une copie de la table à l'instant où le recordset a été créé. Les enregistrements ne sont pas modifiables.

- **Dynamique :** Ouvre un jeu d'enregistrements de type dynamique. Il présente les mêmes caractéristiques que le jeu d'enregistrements de type feuille de réponse dynamique (dynaset) mais ici pour une utilisation avec ODBC (traités plus bas).

- **Forward Only :** Ouvre un jeu d'enregistrements de type "en avant seulement". C'est à dire que le jeu d'enregistrements ne peut – être parcouru qu'une seule fois par l'avant (pas de retour possible). Utile par exemple pour charger un contrôle liste. De ce fait ce mode d'ouverture est intéressant pour sa rapidité de chargement.

Définition du type de recordset avec le contrôle "Data" :

Seul les types "dynaset", "snapshot" et "table" sont disponibles avec le Data Control. Il suffit de sélectionner le type choisit dans la propriété "RecordsetType" de la fenêtre de propriétés.

Définition du type de recordset avec le modèle D.A.O :

Avec le modèle D.A.O, tous les types sont disponibles. Il suffit de choisir la constante correspondante au type choisit et de l'écrire dans l'emplacement "type" de l'instruction ci-dessous lors de l'ouverture du recordset.

Set *PersonneRst* = *MaBase*.OpenRecordset (*source*, ***type***, *options*, *lockedits*)

Tableau des constantes "type" :

Constante	**Libellé**
dbOpenTable	Table
dbOpenDynamic	Dynamique
dbOpenDynaset	Feuille de réponse dynamique
dbOpenSnapshot	Statique
dbOpenForwardOnly	En avant seulement

Définition du type de recordset avec le modèle A.D.O :

Voici les types disponibles avec le modèle ADO, il faut écrire la constante correspondante dans l'emplacement "Cursor type" de l'instruction suivante, c'est à dire lors de l'ouverture :
PersonneRst.Open (source, nom connexion, **cursor type**, Lock type)

Tableau des constantes "cursor type" :

Constante	**Libellé**
dbOpenDynamic	Dynamique
dbOpenKeyset	Feuille de réponse dynamique
dbOpenStatic	Statique
dbOpenForwardOnly	En avant seulement

VERROUILLAGE DES JEUX D'ENREGISTREMENTS

Dans un environnement Multi-Utilisateurs, lorsque l'on ouvre un jeu d'enregistrements, on peut choisir un mode de verrouillage. Cela permet d'empêcher les autres utilisateurs de modifier, en même temps que vous, le même enregistrement. Il existe plusieurs modes de verrouillage :

- Verrouillage pessimiste : l'enregistrement est verrouillé tout de suite, dès lors que l'on exécute la méthode "Edit".
- Verrouillage optimiste : l'enregistrement est verrouillé à la dernière minute, dès lors que l'on exécute la méthode "Update".
- Verrouillage total : Aucune modification n'est autorisé par les utilisateurs.

Mise en œuvre du verrouillage avec le contrôle "Data" :

Mise en œuvre impossible avec le contrôle "Data".

Mise en œuvre du verrouillage avec le modèle D.A.O :

Avec le modèle D.A.O, plusieurs modes de verrouillage sont disponibles. Il suffit de choisir la constante correspondante au mode de verrouillage choisit et de l'écrire dans l'emplacement "lockedits" lors de l'ouverture du recordset.
Remarque : si vous n'utilisez pas les options (options que je ne détaillerai pas) laissez l'emplacement des option vide pour renseigner la constante "Lockedits".
Exemple : la déclaration s'effectue de la manière suivante :

Set *PersonneRst* = *MaBase*.OpenRecordset (*source*, *type*, *options*, ***lockedits***)

Si vous ne renseignez ni le type, ni les options, vous écrirez l'instruction en laissant des paramètres omis mais en laissant les virgules, de la manière suivante :

Set *PersonneRst* = *MaBase*.OpenRecordset (*source*, , , ***lockedits***)

Tableau des constantes "lockedits" :

Constante	**Libellé**
dbReadOnly	Lecture seule
dbPessimistic	Verrouillage pessimiste
dbOptimistic	Verrouillage optimiste
dbOptimisticValue	Avec O.D.B.C
dbOptimisticBatch	Mise à jour par lots

Mise en œuvre du verrouillage avec le modèle A.D.O :
Voici les modes de verrouillages disponibles avec le modèle ADO, il faut écrire la constante correspondante dans l'emplacement "Lock type" de l'instruction suivante :

PersonneRst.Open (source, nom connexion, cursor type, **Lock type**)

Tableau des constantes "Lock type" :

Constante	**Libellé**
adLockOptimistic	Verrouillage optimiste
adLockPessimistic	Verrouillage pessimiste
adLockReadOnly	Lecture seule
adLockBatchOptimistic	Mise à jour par lots

UTILISATION DE O.D.B.C (OPEN DATA BASE CONNECTIVITY)

Le fonctionnement de ODBC repose sur le modèle ADO. Par conséquent, il faut ajouter la même bibliothèque d'objets : "Microsoft ActiveX Data Objects Library 2.1". Pour l'ajouter, cliquez sur "Références" dans le menu contextuel "Projet" et cochez la bonne bibliothèques.

1 – CREATION D'UN DSN

Pour établir une connexion ODBC avec une base de données, il faut créer un DSN permettant d'accéder à la base de données. Pour créer un DSN, il faut lancer l'application ODBC 32 qui se situe dans le panneau de configuration de Windows.

Il existe 3 types de DSN :

- **DSN utilisateur :** DSN utilisable uniquement par un utilisateur (l'utilisateur en cours) et uniquement sur un seul ordinateur.
- **DSN système :** Utilisable par tous les utilisateurs utilisant un même ordinateur et uniquement cet ordinateur.
- **DSN fichier :** Utilisable par tous les utilisateurs sur tous les ordinateurs disposants des mêmes pilotes.

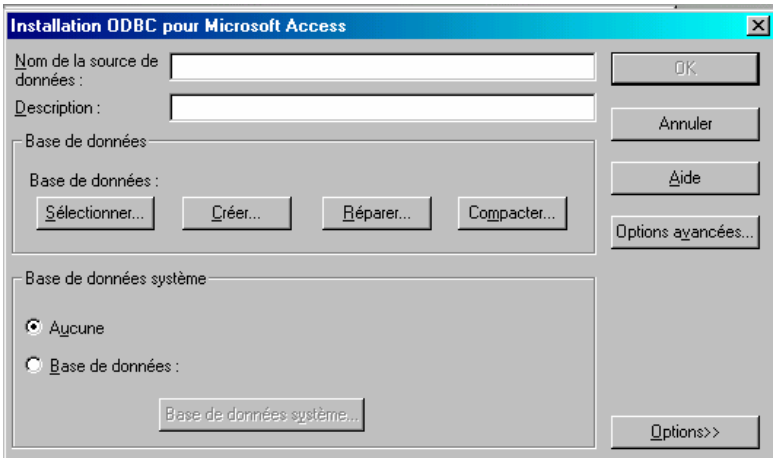
Lorsque l'on crée un DSN utilisateur ou un DSN système, il est utilisable sur le seul ordinateur où il a été créé et est donc sauvegardé sur cet ordinateur. Par contre un DSN fichier est utilisable sur plusieurs ordinateurs donc il est sauvegardé sous forme d'un fichier de type *.dsn qui peut être placé sur toutes les machines devant utiliser ce DSN.
Les procédures de création des trois types de DSN sont identiques :

1 - Lancer l'application ODBC 32 qui se situe dans le panneau de configuration de Windows,
2 - Choisir le type de DSN en cliquant sur l'onglet correspondant au type voulu,
3 - Cliquer sur "Ajouter"
4 - Choisir le pilote voulu (ici **Microsoft Access Driver (*.mdb)**),
5 - Cliquer sur "Suivant",

Etapes supplémentaires pour le DSN fichier :

6 - Choisir un chemin et un nom pour enregistrer le DSN dans un fichier *.dsn
7 - Cliquer sur "Suivant",
8 - Cliquer sur "Terminer".

La fenêtre de création du DSN apparaît :



Une fois les informations remplies, cliquez sur "Ok" : le DSN est créé.

Remarques : pour un DSN fichier, il faut créer une nouvelle base donc on ne peut pas choisir une base de données existante.

Exemple : on a créé un DSN nommé "Bdsalle".

2 – Déclaration d'une connection ODBC

On déclare une connexion. Dans l'exemple suivant on appel la connexion "MaConnection"
Dim MaConnection **As New ADODB.Connection**

3 – Déclaration des recordsets

(Voir la section "Déclaration des recordsets en ADO" puisque c'est ce modèle qui est utilisé avec ODBC).

4 – Ouverture du DSN

Par la ligne de codes suivantes :

MaConnection**.Open "DSN=**NomDsn**"**
MaConnection est le nom déclaré pour la connexion.
NomDsn est le nom que l'on a donné au DSN.

Exemple : MaConnection.Open "DSN=MaBase"

5 – Manipulation des enregistrements

Toutes les opérations sur les jeux d'enregistrements avec ODBC utilisent le modèle ADO. Reportez vous à la section "Accès aux données avec le modèle ADO" pour avoir les méthodes de manipulation des enregistrements.

NOTIONS DE CLIENT / SERVEUR SOUS VB

1 – Le client / serveur d'applications

La notion de client / serveur fait bien sûr appel à la notion de client et à la notion de serveur, chacun ayant son propre rôle. L'exemple le plus simple de client / serveur est celui du client / serveur d'applications.

Exemple : Un utilisateur saisit une chaîne de caractères dans une interface graphique Visual Basic. On veut sécuriser la saisie en vérifiant l'orthographe de la chaîne de caractères. Pour cela, le programme Visual Basic transmet la chaîne de caractères à un logiciel de traitement de texte, comme Microsoft Word, afin d'en utiliser le correcteur orthographique. De cette manière, le client (l'application Visual Basic) demande l'exécution d'une tache au serveur (le logiciel Word). Celui-ci exécute la tache et renvoie le résultat, c'est à dire la chaîne corrigée, au client (l'application Visual Basic).
C'est ce que l'on appel le client / serveur d'applications. Client et serveur sont tous deux des composants logiciel. Ce type de client / serveur est utile lors des développements car il permet de ne pas re-développer des ressources déjà existantes mais de les exploiter.

2 – Le "Vrai" client / serveur

On admet que le concept expliqué ci-dessus est une première approche du client / serveur. Cependant le vrai modèle client / serveur met en scène une application cliente et une application serveur résidantes toutes deux sur des machines différentes.
Par exemple : un ordinateur A sur lequel s'exécute une application Visual Basic demande l'exécution d'une requête qui est stockée sur un autre ordinateur B. Le client (l'ordinateur A) demande l'exécution de la requête par le moteur de base de données du serveur (l'ordinateur B). Ce dernier exécute la requête et renvoi le résultat au client (l'ordinateur A). La requête résidante sur le serveur (l'ordinateur B) est appelée "une procédure stockée".

3 - Remarques sur le "client / serveur"

La création de programmes serveur nécessite la version professionnelle ou entreprise de VB5. Par contre, la création de la partie cliente peut s'entreprendre à partir de toute version de VB.

L'utilisation des bases de données dans des applications client / serveur nécessite la mise en œuvre d'une bonne sécurisation des accès aux données. Cette sécurisation passe avant tous par l'utilisation des options de verrouillage des recordsets (expliquées dans la section "verrouillage des recordsets").

Il peut également être nécessaire d'utiliser les droits d'accès pour gérer l'accès aux base de données (voir l'ouverture d'une base de données en ADO avec les propriétés "username" et "password" dans la section "utilisation du modèle ADO").

Dans les applications lourdes, il est préférable d'utiliser les stratégies de sauvegardes et de synchronisation des données (comme les liaisons et les réplicats) afin d'éviter les incidents et d'optimiser les applications.

Une bonne gestion d'erreurs dans l'application client / serveur est indispensable tant pour la partie cliente que pour la partie serveur. En effet la complexité de ces applications est source d'erreurs (exemple : un programme demande l'ouverture d'une base de données distante qui entre temps aurait été supprimée).

UTILISATION DES PROCEDURES STOCKEES

Remarque : pour utiliser les procédures stockées dans Visual Basic, il ne faut pas nommer vos requêtes Access avec des espaces, utiliser le caractère "_" pour les remplacer.

PROCEDURE STOCKEE NON PARAMETREE :

La ligne de commande est la même que pour ouvrir n'importe qu'elle autre jeu d'enregistrements (table ou requête) :

MonRecordset.Open "NomRequête", "NomConnection", **AdOpenForwardOnly, adCmdStoredProc**

Infos :

- MonRecordset est le nom du jeu d'enregistrement,
- NomRequête est le nom de la requête dans Access,
- NomConnection est le nom de la connection ouverte,
- AdOpenForwardOnly signifie qu'on ouvre le recordset en mode "en avant seulement",
- adCmdStoredProc est la constante qui spécifie que l'on stock une procédure stockée.

Exemple : **MonRecordset.Open** "Nom_des_fournisseurs", "MaConnection", **AdOpenForwardOnly, adCmdStoredProc**

Le résultat est stocké dans un recordset tous à fais ordinaire. Les méthodes de manipulation des enregistrements sont donc ceux de ADO. Reportez vous donc à la section "Modèle ADO" pour plus d'informations sur les manipulations d'enregistrements.

PROCEDURE STOCKEE PARAMETREE :

Situation : On dispose d'une application Visual Basic (Cliente) et d'une base de données Access (Serveur). L'application Visual Basic demande l'exécution de la requête à Access. La base de données contient une requête encore appelée "procédure stockée". Il s'agit d'une requête paramétrée, c'est – à – dire qu'elle attend un paramètre de sélection (pour la clause 'Where'). Ce paramètre est fournit par l'application Visual Basic car il est envoyé à la base de données lors de la demande d'exécution de la requête.

Selon que l'on souhaite ou non installer l'application Visual Basic et la base de données Access sur des ordinateurs différents, on va utiliser ADO ou ODBC (sachant que les deux permettent l'utilisation des base de données distantes mais ODBC peut être plus intéressant dans certains cas de client / serveur).

Dans cet exemple, l'application VB et la base de données Access seront installés sur le même poste. On va donc se contenter d'étudier la procédure stockée avec le modèle ADO.

1 – Ouverture d'une base de données

La première chose à faire est de déclarer la connexion à la base dans le programme.
Dim MaConnection **as new ADODB.Connection** (MaConnection est le nom que je donne à la connexion).

2 - Déclaration d'un Recordset qui va contenir le résultat de la requête (ou procédure stockée)

Dim Procédure **As New ADODB.Recordset** (Procédure est le nom que je donne au jeu d'enregistrements).

3 - Déclaration d'un objet Command

C'est l'objet Command qui va permettre de demander l'exécution de la procédure stockée (ou requête) grâce à sa méthode ".Execute"
Dim CMD **As New ADODB.Command** (CMD est le nom que je donne à l'objet Command).

4 - Préciser la connection à laquelle appartient la requête

CMD**.ActiveConnection** = MaConnection
(CMD est le nom de l'objet Command et MaConnection celui de la connexion).

5 - Préciser le nom de la requête

CMD**.CommandText =** "NomRequête"
CMD est le nom de l'objet Command, NomRequête est le nom de la requête dans la base de données Access.

6 - Exécution de la procédure stockée paramétrée

Ligne de commande :
Set Procédure = CMD**.Execute**(recordsaffected, paramètres)

C'est la méthode ".execute" de l'objet Command qui permet de demander l'exécution de la requête au serveur.
Cette méthode accepte des paramètres :
Recordsaffected : ne pas renseigner se paramètre, mais laisser la virgule.
Paramètres : On précise les paramètres à cet endroit.

Le résultat de la requête sera stockée dans le recordset nommé "Procédure" et pourra être utilisé comme n'importe quel jeu d'enregistrements de type table ou requête. Pour plus d'informations sur la manipulation des jeux d'enregistrements, reportez vous à la section "Modèle A.D.O"
CMD est le nom de l'objet Command.

Exemple : **Set** Procédure = CMD**.Execute**(, CmbFournisseur.Text)
CmbFournisseur est une zone de liste modifiable (ComboList). Ici le paramètre est la ligne sélectionnée dans le contrôle ComboList grâce à la propriété ".text"

7 – Manipulation des enregistrements

Le résultat est stocké dans un recordset tous à fais ordinaire. Les méthodes de manipulation des enregistrements sont donc ceux de ADO. Reportez vous donc à la section "Modèle ADO" pour plus d'informations sur les manipulations d'enregistrements.

LES FICHIERS EN VB

Il y a trois types d'accès aux fichiers en Visual Basic.

- Séquentiel,
- Binaire,
- Direct.

L'ACCES SEQUENTIEL

Définition : Dans un fichier séquentiel, les enregistrements sont écrits et donc lus lignes par lignes (c'est à dire lecture jusqu'à ce que le pointeur rencontre le retour chariot (Code Ascii 13) + saut de ligne (Code Ascii 10). Pour accéder à un enregistrement, il faut lire tous le fichier depuis le premier enregistrement jusqu'à l'enregistrement recherché : l'accès direct est impossible.

1 - Ouverture d'un fichier séquentiel

Exemple : on ouvre un fichier nommé "MonFichier.txt".

```
'FreeFile indique le numéro du prochain fichier libre en mémoire
NumFich = FreeFile
Open "C:\MonFichier.txt" for OutPut as #NumFich'ouverture en écriture (le fichier précédent est effacé)

NumFich = FreeFile
Open "C:\MonFichier.txt" for Append as #NumFich 'ouverture en ajout (enregistrements ajoutés en fin de fichier)

NumFich = FreeFile
Open "C:\MonFichier.txt" for InPut as #NumFich 'ouverture en lecture (ne peut être ouvert qu'en un exemplaire)
```

2 - Ecriture dans un fichier séquentiel

Possible uniquement si le fichier est ouvert en mode écriture ou ajout. Ajout à la fin du fichier avec l'instruction Print.

```
Print #NumFich, "Chaîne de caractères"          'Ecriture d'une chaîne dans le fichier
        'ou
Print #NumFich, variable                        'Ecriture du contenu d'une variable dans le fichier
```

3 - Lecture d'une ligne

Grâce à l'instruction "Line",
'ou
Grâce à l'instruction "Line Input" si les enregistrements contiennent des virgules.

```
Line Input #NumFich, Ligne          'Lecture d'une ligne (de manière séquentielle, on est donc obligé de lire toutes les lignes de la première à la dernière. Charge au programmeur de stocker chaque ligne au fur et à mesure qu'elles sont lues).
```

4 - Affectation des valeurs lues à des variables

Grâce à l'instruction "Input" :

```
Input #NumFich, var1, var2          'Affectation de la première valeur de la ligne à la variable "var1" puis la seconde à la variable "var2" …
```

5 - Lecture séquentielle jusque la fin du fichier

```
While Not Eof(NumFich)              'Tant qu'on a pas atteint la fin du fichier
        Line Input #NumFich, Ligne      'Lire la ligne
Wend
```

6 - Fermeture d'un fichier séquentiel

```
Close #NumFich                      'Fermeture
```

L'ACCES DIRECT (ou Aléatoire)

Définition : Tous les enregistrements composant ce type de fichier sont de tailles identiques ce qui permet de ne pas lire tous le fichiers pour accéder à un enregistrement. L'accès est donc plus rapide que l'accès séquentiel.

1 - Ouverture d'un fichier à accès direct

Exemple : on ouvre un fichier nommé "MonFichier.txt".

```
Open "C:\MonFichier.txt" For Random as #NumFich Len=Len ( VarEnreg )
    'Ouverture du fichier
```

VarEnreg est la variable d'enregistrement.

2 - Définition d'enregistrements par variable personnalisé.

```
Type PERSONNE                       'Définition
    Nom as string
```

```
        Prenom as string
        Age as integer
End Type
```

Déclaration de la variable d'enregistrement avec le type de données personnalisé PERSONNE

```
Dim VarEnreg as PERSONNE                                    'Déclaration
```

3 – Lecture d'enregistrements

Get #NumFich, Numéro d'enregistrement, VarEnreg
Exemple : Get #NumFich, 5, VarEnreg 'Stockage de l'enregistrement n° 5 dans variable VarReng

4 – Ecriture d'enregistrements

Put #NumFich, Numéro d'enregistrement, VarEnreg
Exemple : Put #NumFich, 5, VarEnreg 'Ecriture dans l'enregistrement n° 5 la variable VarReng

5 – Utilisation des champs

Ecriture : VarEnreg.Champ1 = valeur
Exemple : VarEnreg.Champ1 = 55

Lecture : Variable = VarEnreg.Champs 1
Exemple : Var1 = VarEnreg.Champ 1

6 - Fermeture d'un fichier à accès direct

```
Close #NumFich                                              'Fermeture
```

L'ACCES BINAIRE

Les données sont lues et écrites octet par octet. Ce sont en fait des suites d'octets.

1 - Ouverture d'un fichier binaire

Exemple : on ouvre un fichier nommé "MonFichier.bin".

```
NumFich = FreeFile
Open "C:\MonFichier.bin" For Binary as #NumFich             'Ouverture du fichier
```

2 – Lecture dans un fichier binaire

BinVar est le nom de la variable déclarée pour stocker les octets lus. Les données lues sont fonction de la taille de la variable. En général on utilise des variables de type chaîne de caractères pour lire.

```
Get #NumFich, , BinVar                    'Lecture
Exemple : Dim Reponse as string
Get #NumFich, 8, Reponse                  'Stockage des octets lus dans "Reponse"
```

3 – Ecriture dans un fichier binaire

```
Put #NumFich, , BinVar                    'Ecriture
```

4 – Détermination de la position dans le fichier

La position courante est actualisée automatiquement à chaque instruction "Get" (lecture).
L'instruction
Seek #NumFich , , position
positionne le pointeur du fichier sur position.

Exemple : Seek #NumFich , , 5 'Place le pointeur sur 5ème caractère

5 – Fermeture du fichier binaire

```
Close #NumFich                            'Fermeture du fichier
```

FONCTIONS UTILES :

```
Filen ("NomFichier")                                    'Retourne longueur d'un fichier
FileCopy "NomFichierSource", "NomFichierDestination"   'Copie du contenu d'un fichier dans un autre
Kill ("NomFichier")                                     'Suppression d'un fichier
```

LECTURE DES FICHIERS MULTIMEDIA

Le langage Visual Basic dispose de contrôles permettant de lire des fichiers de type média comme des musiques et des vidéos. Attention, il existe trois contrôles différent pour la lecture des fichiers multimédia. L'utilisation de ces contrôles est proche mais pas identique. Il faut prendre garde de ne pas mélanger les propriétés et méthodes de ces contrôles car elles ne sont pas toutes identiques.

Ces trois contrôles sont :

- Le MultimédiaMci,
- Le contrôle ActiveMovie,
- Le contrôle Windows Media Player.

Astuce : si vous désirez développer une application multimédia, il est conseillé d'utiliser une base de données contenant toutes les informations sur les fichiers multimédia à lire (nom, extension et chemin) afin de ne pas les inscrire dans le code source et d'aller les chercher dans la base. Cela est judicieux car lorsque l'on modifiera un nom de fichier ou que l'on en supprimera un, on ne devra pas re-modifier le code source et recompiler, il suffira simplement de modifier les informations dans la base de données. Une bonne gestion d'erreurs est également recommandée.

Le contrôle "MULTIMEDIA MCI"

Pour utiliser ce composant, il faut ajouter le composant "Microsoft Multimédia Control 6.0". Pour cela cliquez sur "Composants" dans le menu contextuel "Projet" et cochez ce composant. Il apparaît ensuite dans la boîte à outil. Placez le contrôle sur la "Form".
D'un point de vue graphique, ce contrôle est composé des boutons de commande standard nécessaires pour l'utilisation des fichiers multimédia (commandes lecture, pause, stop …). Si vous désirez utiliser ces boutons, vous n'aurez pas à charge de programmer les fonctionnalités du player (lecture, pause …) car toutes ces commandes sont intégrés dans le contrôle, il vous faudra quand même préciser les types de fichiers, les noms … et il sera quand même nécessaire d'ouvrir le contrôle par la commande "open", voir ci dessous.
Si par contre, dans un souci de développer une interface graphique plus originale vous ne souhaitez pas utiliser les boutons par défaut, il est toujours possible de créer vos propres boutons et de les programmer.

Voici les étapes chronologique de l'utilisation du contrôle "Multimédia Mci"
Exemple : on nomme un contrôle multimédia "MultimédiaMci ".

1 - Définition du type de fichier à ouvrir

MultimédiaMci.DeviceType = "type"
Exemple : MultimédiaMci.DeviceType = "WaveAudio"

Voici le tableau des chaînes de caractères correspondantes aux types :

Type	Chaîne correspondante
Wav	WaveAudio
Avi	AVIVideo
Mpeg	MpegVideo
Cda (CD audio)	CDAudio
Mov	MMMovie

2 – Définition du fichier média à ouvrir

MultimédiaMci.filename = "chemin + nom + extension"
Exemple : MultimédiaMci.filename = " C:\Windows\MEDIA\Logoff.wav"

3 – Ouverture du contrôle Multimédia

MultimédiaMci.Command = "open"

4 – Commandes d'utilisation du fichier multimédia (grâce à la propriété Command)

```
MultimédiaMci.Command = "play"      'Lecture
MultimédiaMci.Command = "pause"     'Pause
MultimédiaMci.Command = "stop"      'Stop
MultimédiaMci.Command = "prev"      'Début du fichier
MultimédiaMci.Command = "record"    'Enregistrement
MultimédiaMci.Command = "eject"     'Ejection du CD si lecture d'un CD
```

5 – Fermeture du fichier multimédia

MultimédiaMci.Command = "close"

6 – Autre commande

MultimédiaMci.visible = false 'Masque le contrôle, utile lorsqu'on utilise pas les boutons par défaut

7 - Remarques sur la lecture des fichiers vidéo:

Le contrôle Multimédia Player ne prend pas automatiquement en charge l'affichage des séquences vidéo. C'est à dire que lorsqu'on lance la commande de lecture d'un fichier vidéo, le fichier est lu en mémoire mais n'est pas affiché à l'écran. C'est au programmeur de l'afficher. Pour cela il faut utiliser un contrôle "PictureBox". C'est la propriété hWndDisplay qui permet d'afficher la vidéo dans le contrôle PictureBox :
MultimédiaMci.hWndDisplay = NomFrame.hWnd
Exemple : MultimédiaMci.hWndDisplay = Vidéofen.hWnd

L'événement "_Done" du Multimédia Player permet d'exécuter les instructions lorsque le fichier multimédia (audio ou vidéo) est terminé, c'est à dire après la lecture. C'est très utile, par exemple lorsque l'on lit des fichiers de type vidéo, on peut choisir de faire afficher une fenêtre au premier plan contenant une PictureBox diffusant la vidéo. Une fois que la vidéo est terminée, cet événement permet de fermer cette fenêtre qui devient inutile si on désire lire un fichier audio.
Dans le cas ou vous diffusez la vidéo dans un contrôle "PictureBox" contenue dans une autre form que celle où est basée le contrôle Multimédia, utilisez l'instruction suivante :
MultimédiaMci.hWndDisplay = NomForm2.NomFrame.hWnd
Exemple : MultimédiaMci.hWndDisplay = FenêtreFrm.Vidéofen.hWnd

Le contrôle "ACTIVE MOVIE"

Pour utiliser ce composant, il faut ajouter le composant "Microsoft Active Movie Control". Pour cela cliquez sur "Composants" dans le menu contextuel "Projet" et cochez ce composant. Une fois ce composant installé, la référence du même nom est cochée automatiquement dans "Références" dans le menu contextuel "Projet". Le contrôle est apparut dans la boîte à outil. Placez le contrôle sur la "Form".

A la différence du "Multimédia Player", ce contrôle prend en charge l'affichage des vidéos grâce à un écran. Le programmeur n'a donc pas à prendre en charge l'affichage des séquences.

Voici les étapes chronologiques de l'utilisation du contrôle " ActiveMovie "
Exemple : on nomme un contrôle multimédia " ActiveMovie ".

1 – Définition du fichier média à ouvrir

ActiveMovie1.filename = "chemin + nom + extension"
Exemple : ActiveMovie1.filename = " C:\Windows\MEDIA\Logoff.wav"

2 – Commandes d'utilisation du fichier multimédia (grâce à des méthodes)

```
ActiveMovie1.Run          'Lecture
ActiveMovie1.Pause  'Pause
ActiveMovie1.Stop         'Stop
```

3 – Autres propriétés

```
ActiveMovie1.FullScreenMode = True (ou False)   'Plein écran ou non, utile pour les vidéo
ActiveMovie1.visible = True (ou False)          'Affiché à l'écran ou non
```

Remarque : ce contrôle n'a pas besoin d'être fermé.

Le contrôle "WINDOWS MEDIA PLAYER"

Pour utiliser ce composant, il faut ajouter le composant "Windows Média Player". Pour cela cliquez sur "Composants" dans le menu contextuel "Projet" et cochez ce composant. Une fois ce composant installé, la référence du même nom est cochée automatiquement dans "Références" dans le menu contextuel "Projet". Le contrôle est apparut dans la boîte à outil. Placez le contrôle sur la "Form".

Ce contrôle, comme le contrôle "ActiveMovie", prend en charge l'affichage des vidéos grâce à un écran. Le programmeur n'a donc pas à prendre en charge l'affichage des séquences. Il est très proche du contrôle "ActiveMovie" mais lui prend en charge les méthodes "Previous" et "Next".

Voici les étapes chronologiques de l'utilisation du contrôle " Windows Média Player "
Exemple : on nomme un contrôle multimédia " MediaPlayer1 ".

1 – Définition du fichier média à ouvrir

MediaPlayer1.filename = "chemin + nom + extension"
Exemple : MediaPlayer1.filename = " C:\Windows\MEDIA\Logoff.wav"

2 – Ouverture du contrôle

```
MediaPlayer1.Open                'Ouvrir
```

3 – Commandes d'utilisation du fichier multimédia (grâce à la propriété Command)

```
MediaPlayer1.Play                'Lecture
MediaPlayer1.Pause                    'Pause
MediaPlayer1.Stop                'Stop
MediaPlayer1.Previous            'Précédent
MediaPlayer1.Next                'Suivant
```

4 – Autres propriétés

```
MediaPlayer1.DisplaySize = mpFullScreen     'Plein écran ou non, utile pour les vidéo
MediaPlayer1.visible = True (ou False)      'Affiché à l'écran ou non
```

Remarque : ce contrôle n'a pas besoin d'être fermé.