Système d'information
Chapitre 2 : MS Access Niveau 2

Konan Marcellin BROU

marcellin.brou@inphb.ci

2024-2025

1

Sommaire

2

Sommaire

- **Objectifs :**
  - Apprendre les modèles d'accès aux données d'Access ;
  - Savoir créer un état complexe sans assistant ;
  - Savoir créer et utiliser des modules ;
  - Savoir attaquer Access avec Visual Basic ;
  - Savoir attaquer Access avec PHP.

3

I. Introduction

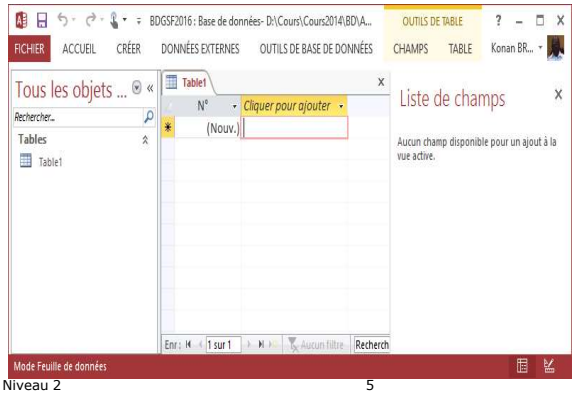
- **1.1. Démarrage d'Access**
  - L'environnement de développement d'Access est celui de Windows.
    - Manipulation des objets liés à cet environnement graphique : souris, fenêtres, menus déroulant….
  - Cliquer sur le bouton Démarrer
  - Sélectionner Toutes les appllications/Microsoft Access

- Cliquer sur l'icône "Base de données vide"

4

I. Introduction

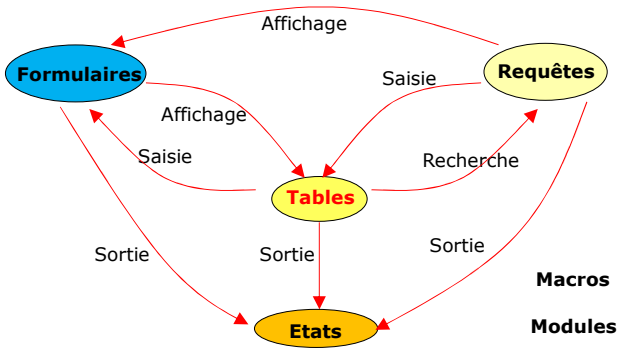
- **Donner un nom à la nouvelle) BD (ex BDGSF2016.accdb et choisir son répertoire de destination.**
- **Cliquer sur le bouton Créer.**



5

I. Introduction

- **1.2. Les éléments de base**
  - **Une BD Access 2007 et + a l'extension .accdb**
    - **Access DataBase**
    - **Elle regroupe dans un seul fichier les objets suivants :**



6

I. Introduction

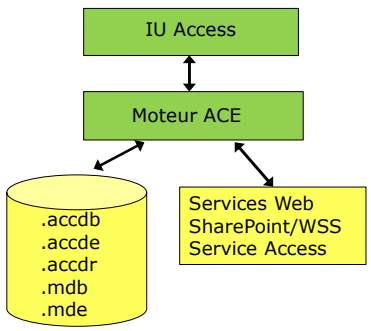
- **Table :**
  - **Les données d'une BD structurées en enregistrements sont réparties entre plusieurs contenants appelés tables ;**
  - **chaque relation est représentée par une table (extension d'une relation).**
- **Formulaire :**
  - **Interface de visualisation ou de saisie de données dans les tables.**
- **Requête :**
  - **Elles permettent de rechercher et de récupérer les données stockées sans les tables répondant à certains critères.**
- **Etats :**
  - **Ils servent à analyser et à imprimer les données stockées dans les tables ou les requêtes.**
- **Macros :**
  - **Commandes permettant d'automatiser certaines tâches (ex. création et impression de bons de livraison et de factures).**
- **Modules :**
  - **Procédures permettant d'automatiser certaines tâches plus complexes non réalisables avec les macros.**

7

II. Architecture du moteur ACE

2.1. Présentation

- **Intégré dans Access 2007, 2010 ou +:**



- **IU Access :**
  - **Interface Utilisateur d'Access**
  - **Responsable de l'interface utilisateur et de tous les moyens par lesquels les utilisateurs affichent, éditent et utilisent des données à travers des formulaires, des rapports, des requêtes, des macros, des assistants, etc.**

8

II. Architecture du moteur ACE

- **2.2. Moteur ACE**
  - **Le moteur Microsoft ACcess Engine fournit les services centraux de gestion de BD :**
    - **Stockage des données : stocker des données dans le système de fichiers.**
    - **Définition des données : créer, éditer ou supprimer des structures pour héberger des données, telles que les tables et les champs.**
    - **Intégrité des données : appliquer les règles relationnelles qui empêchent la corruption des données.**
    - **Manipulation des données : ajouter, éditer, supprimer ou trier des données existantes.**
    - **Récupération des données : récupérer des données dans le système à l'aide de SQL.**
    - **Chiffrement des données : protéger les données contre une utilisation non autorisée.**
    - **Partage des données : partager les données dans un environnement réseau multiutilisateur.**
    - **Publication de données : travailler dans un environnement Web client ou serveur.**

II. Architecture du moteur ACE

- **Importation, exportation et liaison de données : travailler avec des données provenant de différentes sources.**

II. Architecture du moteur ACE

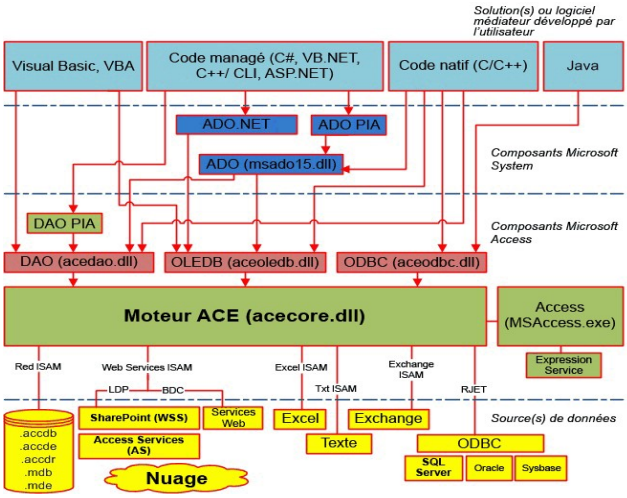
- **2.3. Le moteur JET**
  - **JET : Joint Engine Technology**
  - **Système qui permet d'extraire et d'enregistrer des données dans des BD utilisateur et système Avant Access 2007.**
  - **Depuis la sortie de Microsoft Windows 2000, JET a été inclus dans le système d'exploitation Windows, puis distribué ou mis à jour avec les composants MDAC (Microsoft Data Access Components).**
  - **A la sortie d'Access 2007, le moteur JET a été abandonné et il n'est plus distribué avec MDAC.**
  - **À la place, Access utilise maintenant un moteur ACE intégré et amélioré.**
  - **Le moteur ACE a une compatibilité descendante complète avec les versions antérieures du moteur JET et il peut donc lire et et écrire les fichiers (.mdb) des versions antérieures d'Access.**

II. Architecture du moteur ACE

- **2.4. Technologies d'accès aux données**
  - **Plusieurs moyens pour manipuler des BD Access.**
  - **Les API et les couches d'accès aux données suivantes sont utilisées pour la programmation Access :**
    - **DAO (Data Access Objects)**
    - **OLE DB (Object Linking and Embedding, Database) ;**
    - **ADO.NET**
    - **Objets ADO (ActiveX Data Object)**
    - **ODBC (Open Database Connectivity).**
  - **Le moteur ACE implémente des fournisseurs pour les trois technologies mentionnées : DAO, OLE DB et ODBC.**
  - **Le fournisseur ACE DAO, le fournisseur ACE OLE DB et le fournisseur ACE ODBC sont distribués avec le produit Access**
    - **Excepté ADO, qui fait toujours partie de Microsoft Windows DAC.**

II. Architecture du moteur ACE

- **Beaucoup d'interfaces de programmation, de fournisseurs et d'infrastructures de niveau système pour l'accès aux données incluant ADO et ADO.NET s'appuient sur ces trois fournisseurs ACE.**
- **La figure suivante montre un diagramme représentant une vue d'ensemble des composants Access.**



13

II. Architecture du moteur ACE

- **Explication**

Nom du fournisseur	Informations de connexion	Langage(s) pris en charge
ACE DAO	Acedao.tlh (généré à partir de acedao.dll); acedao.dll	C++
	Set db = CurrentDb() S'exécute dans l'environnement VBE	VBA
ACE OLE DB	Microsoft.ACE.OLEDB.12.0 <Atldbcli.h> et <Atldbsch.h>; Aceoledb.dll	C++
ADO.NET	Microsoft.ACE.OLEDB.12.0 avec System.Data.OleDb ;	C#
	Microsoft.ACE.OLEDB.12.0 Importe System.Data.OleDb Importe System.Console	Visual Basic.NET

Nom du fournisseur	Informations de connexion	Langage(s) pris en charge
ACE DAO	Acedao.tlh (généré à partir de acedao.dll); acedao.dll	C++
	Set db = CurrentDb() S'exécute dans l'environnement VBE	VBA
ACE OLE DB	Microsoft.ACE.OLEDB.12.0 <Atldbcli.h> et <Atldbsch.h>; Aceoledb.dll	C++
ADO.NET	Microsoft.ACE.OLEDB.12.0 avec System.Data.OleDb ;	C#
	Microsoft.ACE.OLEDB.12.0 Importe System.Data.OleDb Importe System.Console	Visual Basic.NET

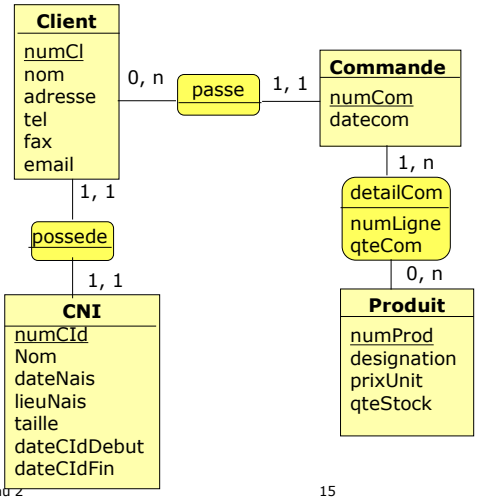
14

III. Les Modules

3.1. Schéma relationnel

- **BD permettant de gérer les commandes.**
- **Règle de gestion**
  - **Un client peut passer ou non au moins une commande (0, n) ;**
  - **Un client possède une seule CNI (1, 1) ;**
  - **Une CNI appartient à un seul client (1, 1) ;**
  - **Une commande est passée par un et un seul client (1,1).**
  - **Une commande contient plusieurs lignes de produits (1, n),**
  - **Un produit peut figurer ou non dans plusieurs commandes (0, n).**

- **Schéma entité/association**



15

III. Les Modules

- **Structure des tables**

Tables	Champ	Type	Taille
Client	numCl	Numérique	Entier long
	nom	Texte court	20 caractères
	adresse	Texte court	30 caractères
	tel	Numérique	Entier long
	fax	Numérique	Entier long
	email	Texte court	50 caractères
	numCId	Texte court	15 caractères
Produit	numProd	Numérique	Entier long
	designation	Texte court	15 caractères
	prixUnit	Numérique	Réel simple
	qteStock	Numérique	Entier
Commande	numCom	Numérique	Entier long
	dateCom	Date/Heure	Date abrégée
	numCl	Numérique	Entier long
DetailCom	numCom	Numérique	Entier long
	numProd	Numérique	Entier long
	numLigne	NuméroAuto	Entier long
	qteCom	Numérique	Entier

Tables	Champ	Type	Taille
CNI	numCId	Texte cours	15 caractères
	nom	Texte court	20 caractères
	dateNais	Date/Heure	Date abrégée
	lieuNais	Texte court	15 caractères
	taille	Numérique	Octet
	dateCIdDebut	Date/Heure	Date abrégée
	dateCIdFin	Date/Heure	Date abrégée

16

III. Les Modules

3.2. Langage VBA

- **Visual Basic pour Application**
- **VBA est le langage commun de tous les produits Microsoft (Word, Excel, Access…)**
- **VBA très proche du langage Visual Basic (VB)**
- **Un module contient du code VBA**
  - **Variables**
  - **Procédures et fonctions**
- **Visual**
  - **Fait référence à la méthode utilisée pour créer l'interface graphique utilisateur (GUI, Graphical User Interface).**
  - **Utilisation d'objet prédéfinis pour construire l'interface graphique**
    - Au lieu de saisir de multiples lignes de code pour décrire l'apparence et l'emplacement des éléments d'interface sur l'écran.
- **Basic**
  - **Fait référence au langage BASIC (Beginners All-Purpose Symbolic Instruction Code)**
  - **Langage le plus utilisé par les programmeurs depuis les débuts de l'informatique.**
  - **Créé en 1964 par John Kemeny et Thomas Kurz du Collège Dartmouth en Californie.**

17

III. Les Modules

3.3. Utilité de Visual Basic

- **Faciliter la maintenance de la BD**
  - **Les macros sont des objets liés aux SGBD Access et indépendants des formulaires qui les utilisent.**
    - Une macro peut être différente d'une version d'Access à une autre.
    - Difficile d'assurer la maintenance d'une BD contenant des macros.
  - **Les procédures événementielles VB sont intégrées dans la définition du formulaire.**
    - Le déplacement d'un formulaire d'une BD à une autre, déplace en même temps les procédures événementielles qui y sont intégrées.
- **Créer ses propres fonctions**
  - **Pour effectuer des calculs complexes pour lesquels il n'existe pas de fonctions prédéfinies.**
- **Créer ou manipuler des objets**
  - **On peut créer et modifier un objet dans son mode Création.**
  - **VB permet de manipuler tous les objets dans une BD, ainsi que la BD elle-même.**

18

III. Les Modules

- **Masquer les messages d'erreur**
  - **Remplacer les messages d'erreur incompréhensibles par des message plus compréhensibles.**
  - **VB permet de détecter l'erreur quand elle survient et, soit afficher un message approprié, soit entreprendre une action.**
- **Exécuter des actions au niveau du système**
  - **Une macro ne permet pratiquement pas d'exécuter des actions à l'extérieur de Microsoft Access.**
    - Seule la macro ExécuterApplication permet lancer une autre application tournant sous Windows depuis une application.
  - **VB permet de :**
    - vérifier si un fichier existe sur le système ;
    - utiliser l'Automatisation ou l'échange dynamique des données (DDE) pour communiquer avec d'autres applications telles que Microsoft Excel,
    - appeler des fonctions dans les DLL (bibliothèques de liaison dynamiques) de Windows.

III. Les Modules

- **Manipuler un enregistrement à la fois**
  - **Les macros ne manipulent les enregistrements que par jeu entier d'enregistrements.**
  - **VB permet de parcourir les enregistrements d'un jeu d'enregistrements un par un et d'accomplir une action sur chaque enregistrement.**
- **Passer des arguments à une procédure VB.**
  - **Lors de la création de macros, on peut définir des arguments pour les actions de macro, on ne peut pas les modifier lors de l'exécution de la macro.**
  - **VB permet de passer des arguments à un code en cours d'exécution :**
    - On peut utiliser des variables comme arguments, impossible dans des macros ;
    - très grande souplesse d'utilisation de de procédures VB.

III. Les Modules

3.4. Notion de classes et d'objet en VB

- **La plupart des composants VB sont disponibles sous forme de classe.**
- **classe**
  - **Type défini par le programmeur**
  - **Rassemble dans une même entité les éléments de données (variables) et les éléments de traitement de ces données (fonctions ou méthodes ou opérateurs).**
  - **Regroupe un ensemble d'objets ayant des propriétés communes.**
- **Objet**
  - **Entité générique manipulable par un ordinateur : une fenêtre, un bouton, un champ de saisie, …**

III. Les Modules

- **Propriétés d'un objet :**
  - **Nom : tout objet porte un nom.**
    - Exemple : "cmdOK" nom d'un bouton.
  - **Classe : tout objet appartient à une catégorie d'objets ou classe. On dit qu'un objet est une instance d'une classe. En VB on distingue les classes suivantes :**
    - la classe des feuilles c'est à dire les fenêtres ;
    - les classes de contrôles, bouton à cliquer, étiquette... ;
    - les classes d'objets spéciaux (le presse-papiers (ClipBoard), le système de débogage (Debug)
    - l'écran (Screen), l'application (App), le moteur de bases de données Jet (DbEngine).
  - **Propriétés : caractéristiques propres à un objet.**
    - Exemple : titre (Text) la hauteur (Height), la largeur (Width) et le nom (Name) d'un objet bouton.
  - **Méthodes : programmes permettant de manipuler un objet.**
    - Exemple : méthodes Show (rend visible) et Hide (cacher) définies sur un objet fenêtre.

III. Les Modules

- **Evénements et procédures d'événement : toute action qui se produit sur un objet est appelé événement.**
  - Le clic sur un bouton produit l'événement Click,
  - La modification d'une zone de texte déclenche l'événement Change.
  - La procédure d'événement est un programme écrit par l'utilisateur qui s'exécutera chaque fois que l'événement se produit.

III. Les Modules

3.5. Les instructions de base

Les identificateurs

- **Mot choisi par le programmeur pour désigner un objet du programme (variable, fonction…).**
- **Caractéristiques :**
  - Commence par une lettre de l'alphabet ;
  - Longueur max : 255 caractères ;
  - Pas de point (.) ou de caractères spéciaux caractérisant les types de données ($, %, #…) ;
  - unique dans son domaine de validité ;
  - Pas de casse.

Les caractères spéciaux

- **Ont une signification particulière en VB**
- **Exemples :**
  - **:** : séparateur d'instruction ;
  - **%** : déclaration d'entier ;
  - **$** : déclaration de chaînes de caractères.

Les mots réservés

- **Identificateurs prédéfinis du langage VB ;**
- **ils ne doivent pas être utilisés comme identificateur dans un programme.**
- **Exemple :**
  - FUNCTION, WHILE, SUB

III. Les Modules

- **Les commentaires**
  - **Un commentaire peut être insérer dans un programme afin de faciliter sa lecture et sa maintenance.**
  - **Il commence par une quote suivi du texte du commentaire.**
  - **Exemple :**
    - 'prix_TTC désigne le prix Toute Taxe Comprise

- **Les constantes**
  - **C'est une donnée dont la valeur reste inchangée tout au long de l'exécution du programme.**
  - **Syntaxe :**
    - CONST identificateur = expression
  - **Exemple :**
    - CONST Pi=3.14, Pi2=PI*PI
  - **Constantes prédéfinies :**
    - VbCrLf : (Carriage return et Line feed) caractère de saut de ligne
    - True : valeur –1
    - False : valeur 0

III. Les Modules

- **Les variables**
  - **Définition**
    - Identificateur utilisé pour désigner une valeur quelconque d'un ensemble donné.
    - Sa valeur peut changer pendant l'exécution du programme.
    - En fait une adresse où long range une valeur.

  - **Déclaration implicite de variable**
    - Pas de déclaration, la réservation mémoire est effectuée dès leur première utilisation.
    - Le type de ces variables dépend du type de l'expression qui leur est assigné.
    - Exemple
      - Omega = 100 * Pi
      - Plante = "Acacia senegal"

III. Les Modules

- **Déclaration implicite de variable**
  - Dans ce cas, la variable nécessite l'une des instructions de déclaration suivantes :

Déclaration	Explication
Dim identificateur [As Type]	Variable de classe automatique
Static identificateur [As Type]	Variable de classe statique
Public identificateur [As Type]	Variable de type public
Global identificateur [As Type]	Variable globale
Private identificateur [As Type]	Variable de type privé

- **La portée de variable**
  - **Portée locale :**
    - variable définie à l'aide des instructions Dim ou Static dans une procédure ;
    - visible qu'à l'intérieur de cette procédure.
  - **Portée globale :**
    - variable définie dans la partie déclaration de n'importe quel module à l'aide des instructions Public ou Global ;
    - visible par toutes les procédures du projet.

III. Les Modules

- **Durée de vie des variables**
  - Temps de présence d'une variable dans la mémoire centrale.
  - Vie limité à la procédure :
    - variables locales définies par l'instruction Dim
  - Vie limitée au projet :
    - variables locales définies par les instructions Static et globales

- **Les types de données**
  - **Un type est constitué par un ensemble de valeurs**

Type	Description
Array	Array.
Boolean	Boolean. (True or False.)
Byte	Byte. (0 à 255.)
Char	Char. (0 à 65535.)
Currency	Currency.
DataObject	DataObject.
Date	Date. (0:00:00 on janvier 1, 0001 à 11:59:59 PM on Decembre 31, 9999.)
Decimal	Decimal. (0 à +/- 79,228,162,514,264,337,593,543,950,335 pas de point décimal; 0 à +/- 7.9228162514264337593543950335 avec 28 places à droite de la décimale; plus petite valeur non nulle +/- 0.0000000000000000000000000001.
Double	Double. -1.79769313486231E+308 à -4.94065645841247E-324 : valeurs négative ; 4.94065645841247E-324 à 1.79769313486231E+308 : les valeurs positive.

III. Les Modules

Type	Description
Empty	Reference Null.
Error	Exception Systeme
Integer	**Integer**. (-2,147,483,648 à 2,147,483,647.)
Long	**Long**. (-9,223,372,036,854,775,808 à 9,223,372,036,854,775,807.)
Null	Object Null.
Object	Remplace le type Variant (n'importe quel type).
Short	**Short**. (-32,768 à 32,767.)
Single	**Single**. -3.402823E+38 à -1.401298E-45 pour valeur <0; 1.401298E-45 à 3.402823E+38 pour valeur >0
String	**String**. (0 à 2 milliards caractères Unicode.)
Variant	**Variant**.
Structure	Ensemble de différentes variables définies par l'utilisateur
UInteger	Unsigned (non signé) Entier codé sur 32 bits pouvant prendre les valeurs 0 à 4 294 967 295.
ULong	Entier codé sur 64 bits :0 à 18 446 744 073 709 551 615
UShort	Entier sur 16 bits 0 à 65 535
SByte	Byte mais signé. Codé sur 1 octet, valeur de -128 à 127

- **Le type Object**
  - Remplace le type Variant.
  - Permet de stocker divers types de données.
  - Les conversions de type sont prises en charge automatiquement par VB.
  - Exemple :

```
Dim varLibre as Object
varLibre = "95" 'Chaîne de caractères 95

varLibre = "Windows " & varLibre 'Chaîne de caractères Windows 95

varLibre = "95" 'Chaîne de caractères 95

varLibre = varLibre + 119 'Valeur numérique 214 = 95 + 119
```


III. Les Modules

- **Les opérateurs**
  - **Opérateurs arithmétiques**

Opérateur	Description
+	addition
-	soustraction
*****	multiplication
/	division
****	division entière
^	puissance
Mod	modulo
Op=	Op ∈ {+, -, *, /, …}

  - **Opérateurs de comparaison**

Opérateur	Description
>	Supérieur à
<	Inférieur à
=	Égal à
>=	Supérieur ou égal à
<=	Inférieur ou égal à
<>	Différent de
Like	Comme la chaîne fournie

- **Exemples :**

```
Dim produit As String
If produit Like "Vis" Then   'produit commence par Vis
…
If produit Like "*Vis" Then  'produit se termine par Vis
…
If produit Like "*Vis*" Then  'produit contient Vis
…
```

III. Les Modules

- **Opérateurs logiques**

Opérateur	Description
Or	Ou inclusif
Xor	Ou exclusif
And	Et
Not	Non
<=Eqv	Équivalence
Imp	Implication

  - Table de vérité

A	B	Not	And	Or	Xor	Eqv	Imp
True	True	False	True	True	False	True	True
True	False	False	False	True	True	False	False
False	True	True	False	True	True	False	True
False	False	True	False	False	False	True	True

- **Opérateurs de chaînage**

Opérateur	Description
&	Concaténation
+	Ancien opérateur

  - Exemple :

Ecole = "INP" & "-HB" donne"**INP-HB**"
Ecole = "INP" + "-HB" donne "**INP-HB**"

III. Les Modules

- **3.6. Structure d'un programme VBA**

```
Option Compare Database
Option Explicit
Private Sub CommandeCommande_Click()
On Error GoTo Err_CommandeCommande_Click
    Dim stDocName As String
    Dim stLinkCriteria As String
    stDocName = "Commande"

    stLinkCriteria = "[numCl]=" & Me![numCl]
    DoCmd.OpenForm stDocName, , , stLinkCriteria
Exit_CommandeCommande_Click:
    Exit Sub
Err_CommandeCommande_Click:
    MsgBox Err.Description
    Resume Exit_CommandeCommande_Click
End Sub
```

- **Explication**

III. Les Modules

- **La partie Option**
  - **Option Compare**
    - Détermine la méthode de comparaison des chaines de caractère
    - OPTIONS COMPARE TEXT : considère qu'un caractère majuscule et égale à son pendant minuscule.
    - OPTION COMPARE BINARY : considère que les caractères sont classés dans leur ordre d'apparence dans la table ASCII. A=65 a un poids inférieur à a=91.
    - de variables par défaut
    - OPTION COMPARE DATABASE : qui ne peut être utilisée que dans Microsoft Access. Elle fournit des comparaisons de chaînes basées sur l'ordre de tri déterminé par l'identificateur de paramètres régionaux de la base de données dans laquelle la comparaison de chaînes est effectuée.
  - **Option Explicit**
    - Force le programmeur à déclarer toutes ses variables.
    - Pas de déclaration Remarque
    - Option Explicit et Option Compare Database doivent précédées toutes les autres instructions.

III. Les Modules

- **L'affectation**
  - **Instruction la plus élémentaire.**
  - **Consiste à affecter à une variable la valeur d'une expression de même type ou de type compatible.**
  - **Syntaxe :**
    - **Identificateur = Expression**
    - identificateur reçoit la valeur de Expression
    - Ranger la valeur de Expression dans identification

III. Les Modules

- **Instruction d'écriture**
  - **Opération qui permet de véhiculer les données de la mémoire centrale vers l'extérieur (ex. écran).**
  - **MsgBox()**
    - Syntaxe :

```
Dim Reponse As integer, Prompt As String
Dim Titre As String, Bouton As Integer
Reponse = MsgBox(Prompt, Bouton, Titre)
```

  - **Avec**
    - Prompt : message d'invite.
    - Titre : titre de la boîte de dialogue d'entrée de données.
    - Bouton : Facultatif. Expression de MsgBoxStyle indiquant le type de boutons à afficher, le style d'icône à utiliser, l'identité du bouton par défaut, ainsi que la modalité du message. (sa valeur par défaut est OKReadOnly).
    - Reponse : contient le code du bouton qui a été enfoncé.
  - **Liste des constantes**

III. Les Modules

Constante	Valeur	Description
OKOnly	0	Affiche le bouton **OK** uniquement.
OKCancel	1	Affiche les boutons **OK** et **Annuler**.
AbortRetryIgnore	2	Affiche le bouton **Abandonner**, **Réessayer** et **Ignorer**.
YesNoCancel	3	Affiche les boutons **Oui**, **Non** et **Annuler**.
YesNo	4	Affiche les boutons **Oui** et **Non**.
RetryCancel	5	Affiche les boutons **Réessayer** et **Annuler**.
Critical	16	Affiche l'icône **Message critique**.
Question	32	Affiche l'icône **Requête d'avertissement**.
Exclamation	48	Affiche l'icône **Message d'avertissement**.
Information	64	Affiche l'icône **Message d'information**.
DefaultButton1	0	Le premier bouton est le bouton par défaut.

Constante	Valeur	Description
DefaultButton2	256	Le 2e bouton est pris par défaut.
DefaultButton3	512	Le 3e bouton est pris par défaut.
ApplicationModal	0	Boîte de dialogue modale, à fermer avant de continuer.
SystemModal	4096	Modal système. Toutes les applications sont interrompues jusqu'à ce que l'utilisateur réponde au message affiché.
MsgBoxHelp	16384	Ajoute le bouton Aide à la zone de message.
MsgBoxSetForeground	65536	Indique la fenêtre de zone de message comme fenêtre de premier plan.
MsgBoxRight	524288	Le teste est aligné à droite.
MsgBoxRtlReading	1048576	Le texte apparaît de droite à gauche sur les systèmes hébraïques et arabes.

III. Les Modules

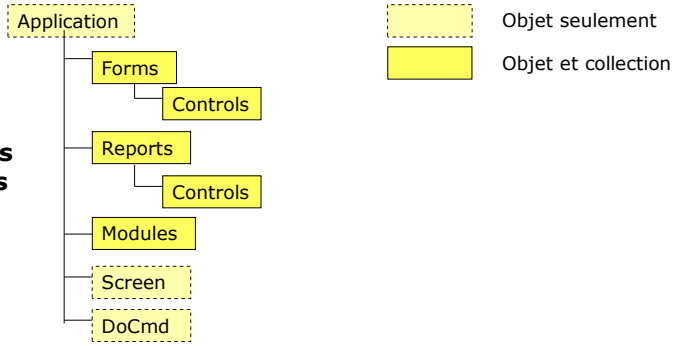
- **Instruction de lecture**
  - **Opération qui consiste à véhiculer les données de l'extérieur (ex. clavier) vers la mémoire de l'ordinateur.**
  - **C'est une affectation particulière.**
  - **InputBox()**
    - Syntaxe :

```
Dim identificateur As String
Dim Prompt As String, Titre As String
identificateur = InputBox(Prompt, Titre)
```

  - **Avec**
    - Prompt : le message d'invite
    - Titre : le titre de boîte de dialogue d'entrée de données

III. Les Modules

- **3.7. Les objets d'Access**
  - **Deux catégories d'objets :**
    - **les objets d'Access (tels que formulaire et états) ;**
    - **les objets d'accès aux données gérés par le moteur de BD tels que table et requête.**
  - **Les objets d'Access**



III. Les Modules

- **L'objet Application**
  - **Contient tous les objets et collections Microsoft Access.**
  - **Utilisez l'objet Application pour appliquer des méthodes ou des paramètres de propriété à toute l'application Microsoft Access.**
  - **la méthode SetOption de l'objet Application peut être utilisée pour définir des options de BD à partir de VB.**

- **Exemple**
  - Activer la case à cocher Barre d'état en dessous de Afficher dans l'onglet Affichage de la boîte de dialogue Options.

```
Application.SetOption "Afficher la barre d'état", True
```

- **La fonction CreateObject:**
  - Permet de créer une instance de la classe Application.
  - Exemple **:**

```
Dim appAccess As Object
Set appAccess = CreateObject("Access.Application.8")
```


III. Les Modules

- **La méthode OpenForm**
  - Méthode de l'objet DoCmd de Microsoft Access permet d'ouvrir un formulaire Microsoft Access à partir de Microsoft Excel.
  - Exemple :

```
appAccess.DoCmd.OpenForm "Commandes"
```

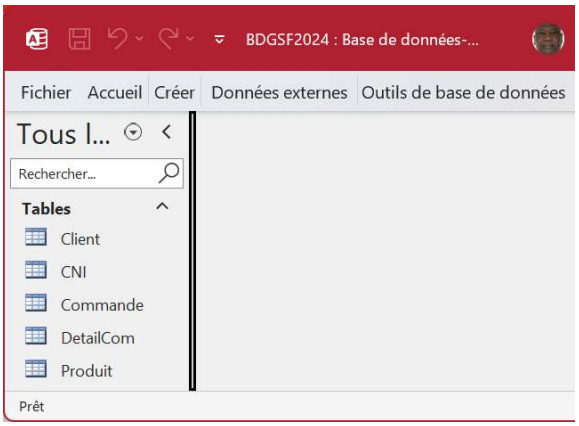
- **Exemple complet**
  - Imprimer certains paramètres de propriété en cours de l'objet Application, et définir une option et quitter l'application en enregistrant tous les objets.

- **Exemple : boites MsgBox et informations**

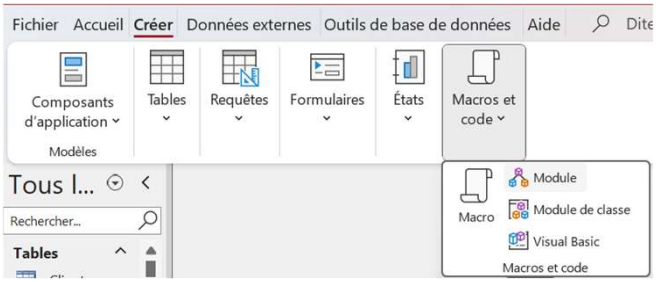
```
Sub InfoApplication()
 Dim Msg, Title, MyString As String
 Dim Style As VbMsgBoxStyle, Response As Integer
 Msg = "Souhaitez-vous continuer?"
 Style = vbYesNo Or vbCritical Or vbDefaultButton2 'Définit les
boutons.
 Title = "Avertissement" ' Définit le titre.
  'Imprime le nom et le type de l'objet actif.
 Debug.Print Application.CurrentObjectName '(table, module...)
 Debug.Print Application.CurrentObjectType
 'Définit l'option Objets masqués sous Afficher dans
 'l'onglet Affichage de la boîte de dialogue Options.
 Application.SetOption "Afficher objets masqués", True
 Response = MsgBox(Msg, Style, Title)
 Debug.Print Response
 'Quitte Microsoft Access, en enregistrant tous les objets
 If Response = 6 Then
  Application.Quit acSaveYes
 End If
End Sub
```

III. Les Modules

- **Création d'un module**
  - **Ouvrir le BD Access BDGSF**



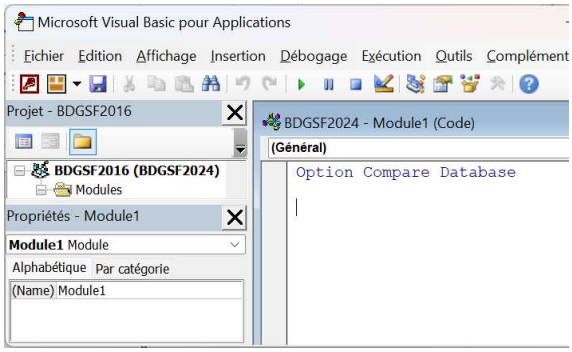
  - **Cliquer sur l'onglet Créer/icône Macro**



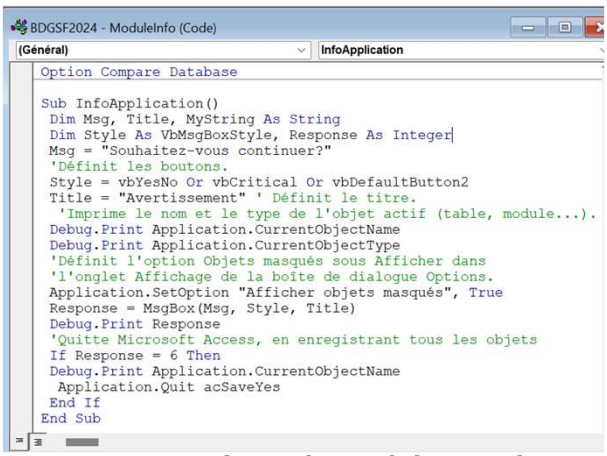
  - **Sélectionner Module**

41

III. Les Modules



  - **Saisir le code précédent**

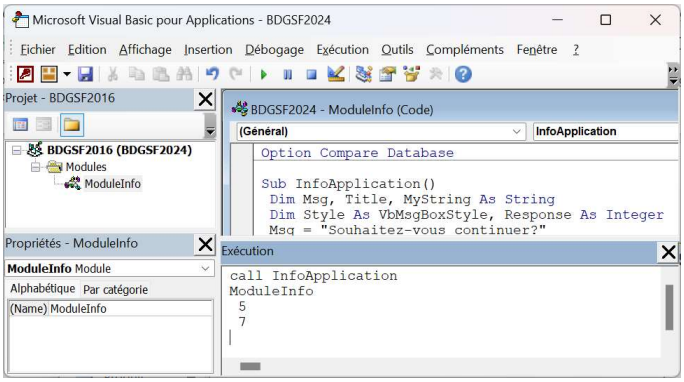


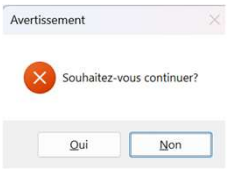
  - **Enregistrer le module sous le nom moduleInfo**

42

III. Les Modules

- **Saisir : call InfosApplication**





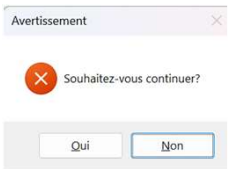
- **Cliquer sur Non**

Nom de l'objet actif : ModuleInfo
Type de l'objet actif : 5
Valeur retournée par MsgBox : 7

- **Appuyer sur la touche Entrée**

III. Les Modules

- **Effacer l'affichage**
- **Réexécuter call InfosApplication**



- **Un clic sur Oui enregistre et ferme l'application**

III. Les Modules

- **Variables objet**
  - **On peut déclarer dans un programme des variables ayant pour type un de ces objets.**
  - **Tous ces objets possèdent des propriétés événementielles déclenchées :**
    - soit par des actions de l'utilisateur (ex. clic sur un bouton de commande) ;
    - soit lors de mise à jour de ces objets (ex. après mise à jour d'un formulaire).

  - **Exemple 1 : programme permettant de lire le numéro d'un client dans le formulaire Client.**
    - Saisir le code suivant :

```
Public Sub objetForm()
  Dim monForm As Form
  Dim monControl As Control, Msg, Title, MyString As String
  Dim Style As VbMsgBoxStyle, Response As Integer

  Set monForm = Forms![Client]
  Set monControl = monForm![numCl]
  Msg = "Numéro du client courant : " & monControl.OldValue
  Response = MsgBox(Msg, vbInformation, "Information")
End Sub
```

    - Enregistrer le module sous ModuleGeneral

III. Les Modules

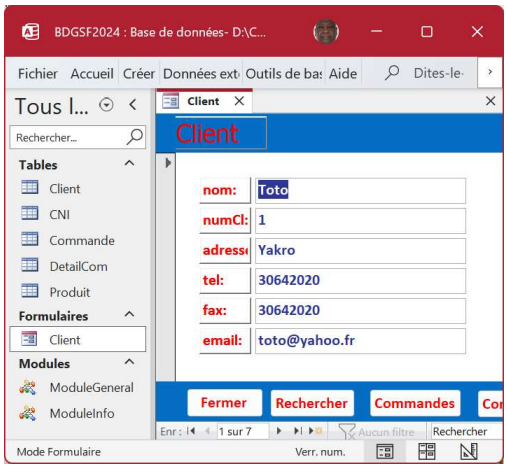
- **Tous les formulaires sont dans la collection Forms**

  Set monForm = Forms![Client]

- **Accès à un contrôle du formulaire**

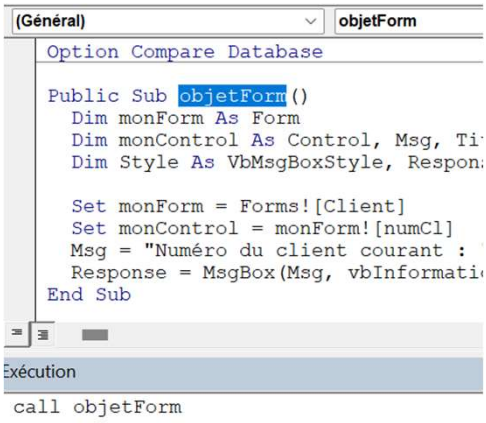
  Set monControl = monForm![numCl]

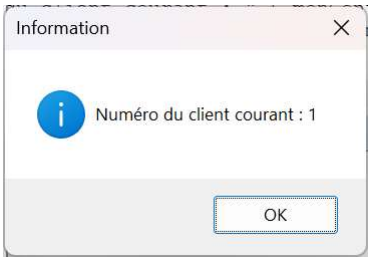
- **Ouvrir le formulaire Client**



III. Les Modules

- **Exécuter la procédure objetForm**





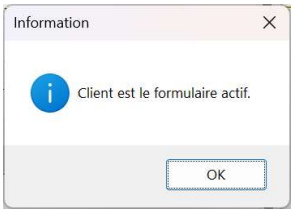
III. Les Modules

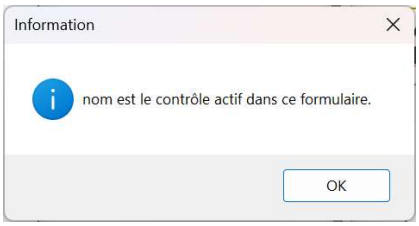
- **Exemple 2 : Afficher le nom du formulaire courant et du contrôle courant.**
  - Saisir le code suivant dansle module ModuleGeneral :

```
Sub ObjetsActifs()
   Dim monForm As Form, monControl As Control, Response As Integer
   'Retourne un objet Form pointant vers un formulaire actif.
   DoCmd.SelectObject acForm, "Client", False
   Set monForm = Screen.ActiveForm
   'Retourne un objet Control pointant vers un contrôle actif.
   Response = MsgBox(monForm.Name & " est le formulaire actif.", _
   vbInformation, "Information")
   Set monControl = Screen.ActiveControl
   Response = MsgBox(monControl.Name & " est le contrôle actif" & _
   " dans ce formulaire.", vbInformation, "Information")
End Sub
```

  - Ouvrir le formulaire Client

- **Exécuter la procédure ObjetsActifs**





III. Les Modules

- **Exemple 3 : Afficher le nom des champs d'une table et leur contenu.**

Nombre de colonnes : maTable.Fields.Count – 1
Nom d'une colonne : maTable.Fields(i).Name
Nom de lignes : maTable.RecordCount – 1

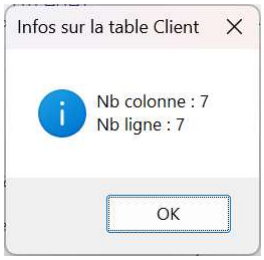
- Ajouter le code suivant au module Module General

```
Public Sub afficheInfosTable(nomTable As String)
Const cheminBD = "D:\Cours\ESMG-
ESPE\ESP\Support\SystemeInformation\BDGSF2024.accdb"
 Dim dB As Database, maTable As TableDef
 Dim titre, message As String
 Dim iLigne, iCol, nbLigne, nbColonne As Integer

 Set dB = OpenDatabase(cheminBD)
 Set maTable = dB(nomTable)
 nbLigne = maTable.RecordCount
 nbColonne = maTable.Fields.Count
 For iCol = 0 To nbColonne - 1
    ch = ch & maTable.Fields(iCol).Name & " "
 Next
 titre = "Infos sur la table " & nomTable
 message = "Nb colonne : " & nbColonne & vbCrLf & "Nb ligne :
" & nbLigne
 Response = MsgBox(message, vbInformation, titre)
End Sub
```


49

III. Les Modules

- **Exécuter la procédure afficheInfosTable**



50

VI. Le modèle ADO

6.1. Présentation

- **ADO (ActiveX Data Object) est un composant ActiveX.**
- **Modèle objet d'accès aux données qui s'appuie sur OLEDB.**
- **Permet de se connecter à tout type de données et d'exécuter des requêtes SQL sur cette base :**
  - **BD Relationnelles (SQL Server, Oracle…) ;**
  - **Autres sources de données non relationnelles telles que des fichiers texte ;**
  - **Des sources de données non Microsoft.**
- **ODBC : Open DataBase Connectivity**
  - **Protocole standard d'accès aux données.**
  - **Il permet à une application de se connecter de la même façon à un ensemble de BD relationnelles différentes.**
  - **Fédère l'accès aux BDR, alors qu'OLEDB fédère l'accès à tous types de données, relationnelles ou non.**
- **Références à ajouter au module VBA :**
  - **Microsoft ActiveX Data Objects 6.x Library**

51

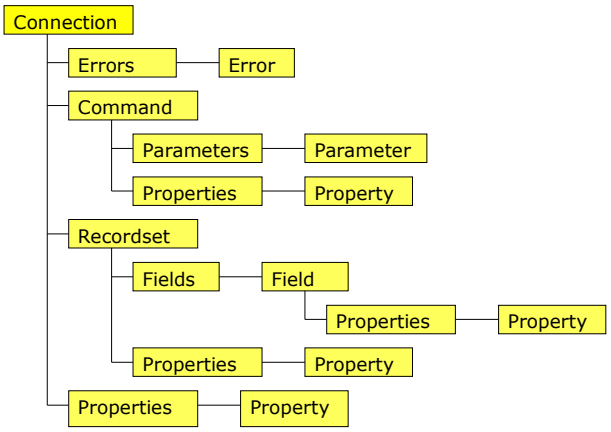
VI. Le modèle ADO

- **Pilote ODBC (ou driver ODBC) :**
  - **DLL (Dynamic Link Library) qui répond à un ensemble de spécifications.**
  - **Implémenter les accès spécifiques à une BD particulière.**
    - driver ODBC pour SQL Server, Oracle, DB2….
  - **S'appuie sur les librairies de bas niveau pour accéder aux BD :**
    - DBLib pour SQL Server
    - SQLNet pour Oracle…
- **OLEDB : Object Linking and Embedding DataBase**
  - **Technologie qui a pour but de résoudre certaines contraintes d'ODBC :**
    - Autorise l'accès à tout type de données ;
    - Permet de gérer l'aspect distribué des sources de données ;
    - Prend en compte les contraintes du Web.
  - **Son but à terme est de remplacer la technologie ODBC.**
  - **ADO est le modèle objet qui permet de simplifier l'accès à cette technologie.**

52

VI. Le modèle ADO

- **6.2. Le modèle de données d'ADO**
  - **Objets les plus significatifs : Connection, Recordset, Command.**



VI. Le modèle ADO

- **Description**

Objet ou Collection	Utilisation
Objet Connection	Permet d'établir les connexions entre le client et la source de données
Objet Command	Permet de réaliser des commandes, telles que des requêtes SQL ou des mises à jour d'une base.
Objet Recordset	Permet de voir et de manipuler les résultats d'une requête
Collection Parameters	Est utilisée lorsque la requête de l'objet Command nécessite des paramètres
Collection Errors	La collection Errors et l'objet Error sont accédés au travers de l'objet Connection, lorsqu'une erreur du fournisseur est générée.
Collection Fields	La collection Fields et l'objet Field sont accédés au travers de l'objet Recordset, une fois que celui-ci contient des données

VI. Le modèle ADO

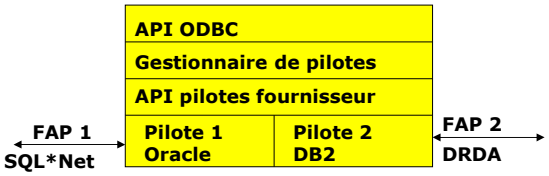
6.3. Création d'une source de données ODBC

- **ODBC**
  - **Open DataBase Connectivity**
  - **Interface de programmation définie au départ pour le langage C permettant l'interopérabilité avec les serveurs de BD.**
    - ODBC définit une librairie de fonctions qui constitue l'interface pour se connecter à un SGBD, exécuter des ordres SQL, et retrouver des résultats.
    - ODBC permet à une application de se connecter de la même façon à un ensemble de BD relationnelles différentes.

- **ODBC Driver**
  - **Un driver ou pilote ODBC est une Dll qui répond à un ensemble de spécifications.**
  - **C'est le driver ODBC qui va implémenter les accès spécifiques à une BD particulière.**
    - Il existe ainsi des drivers ODBC pour des bases SQL Server, Oracle, DB2….
  - **Le driver s'appuie sur les librairies de bas niveau pour accéder aux bases :**
    - Par exemple, DBLib pour SQL Server, SQL*Net pour Oracle….

VI. Le modèle ADO

- **Architecture d'ODBC**



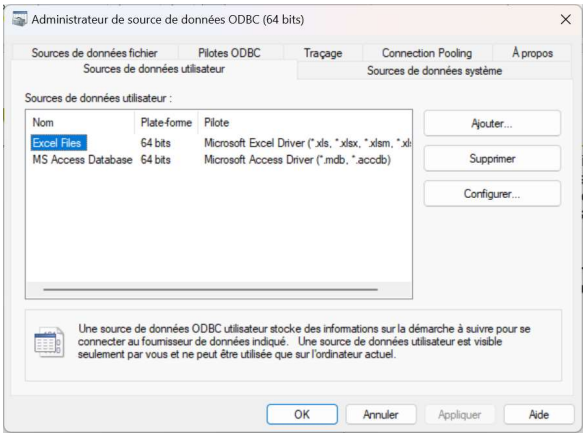
  - **API d'ODBC :**
    - Couche de réception des requêtes ODBC.
  - **Gestionnaire de pilote :**
    - Couche de gestion des pilotes (à quel pilote passer une requête?) selon un format standard pour les pilotes ODBC.
  - **Pilote i :**
    - chaque pilote de BD encapsule les requêtes dans le format propre à chaque fournisseur de BD.
  - **FAP i (Format And Protocol) :**
    - Structure des messages et codage des données pour un SGBD.

VI. Le modèle ADO

- **Noms de source de données**
  - **Ou Data Source Name (DSN)**
  - **Permet de manipuler une BD dans une application**
  - **Plusieurs types de DSN :**
    - **DSN Système** : permet à tous les utilisateurs connectés à un serveur défini de se connecter à une BD ;
    - **DSN Utilisateur** : limite la connectivité à la BD à un utilisateur spécifique disposant des informations d'authentification nécessaires.
    - **DSN fichier** : fichier texte, fournit un accès à plusieurs utilisateurs et se transfère aisément d'un serveur à un autre par copie.
  - **Les noms de DSN Utilisateur et Système se situent dans le registre de Windows NT.**

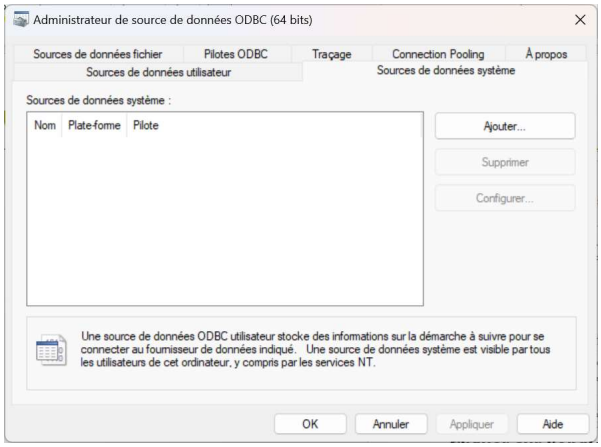
VI. Le modèle ADO

- **Configuration d'une DSN System de BD Access**
  - **Saisir ODBC dans la zone de recherche**
  - **Sélectionner Source de données système**

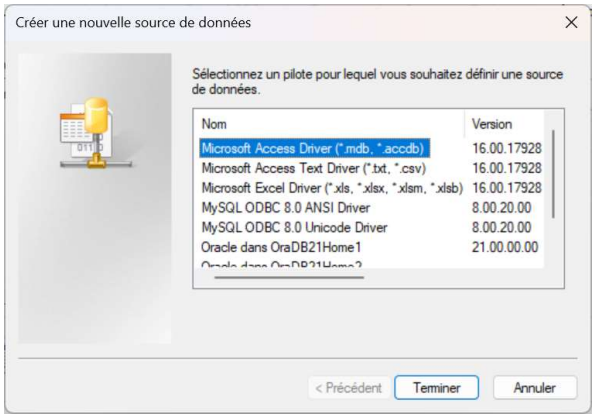


  - **Cliquer sur l'onglet Source de données système**

VI. Le modèle ADO



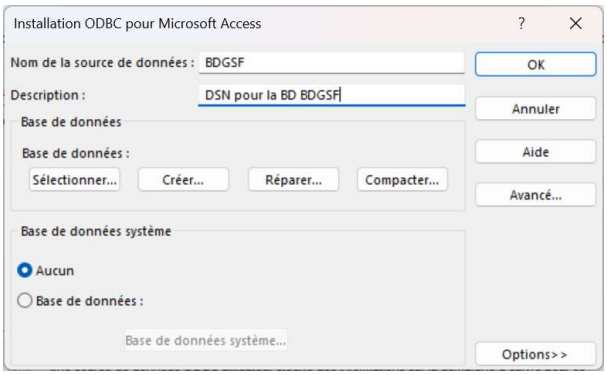
- **Cliquer sur Ajouter**



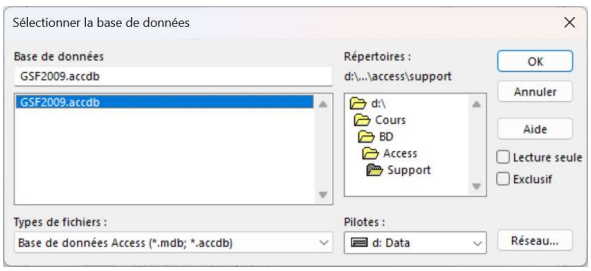
- **Sélectionner Microsoft Acces Driver (*.mdb, *.accdb) et cliquer sur Terminer**

59

VI. Le modèle ADO



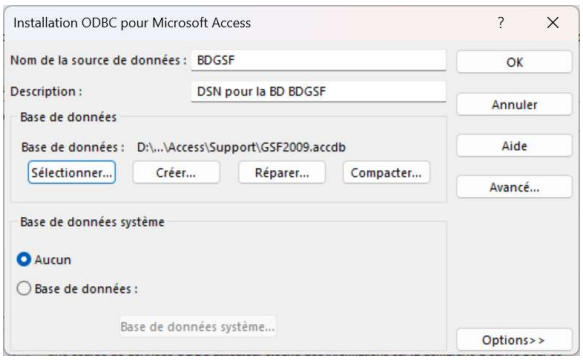
- **Saisir le nom de la source de données" (BDGSF)**
- **La description de la BD est facultative**
- **Cliquez sur le bouton Sélectionner**



- **Sélectionner l'emplacement de la BD sur le disque dur de l'ordinateur**
- **Cliquer sur le bouton Ok**

60

VI. Le modèle ADO



- **Cliquer sur le bouton Ok**

61

VI. Le modèle ADO

- **6.4. Etablir une connexion**
  - **On utilise l'objet Connection.**
  - **Une connexion à une BD se définit par :**
    - **l'hôte sur lequel se trouve la BD ;**
    - **le nom de la BD ;**
    - **le nom de l'utilisateur ;**
    - **le mot de passe.**
  - **L'ensemble de ces champs est appelé chaîne de connexion.**
  - **Les champs "hôte" et "nom de la BD" sont définis dans le programme ou dans une DSN (Data Source Name).**
    - **Une DSN se configure dans le panneau de configuration avec l'outil Source de données (ODBC).**

62

VI. Le modèle ADO

- **Connexion à la BD sans un DSN**
  - **Déclaration de la variable**

```
Dim cnx As New ADODB.Connection
ou
Dim cnx As ADODB.Connection
Set cnx = New ADODB.Connection
```

  - **Définition du pilote de connexion**

```
cnx.Provider = "Microsoft.ACE.OLEDB.12.0"
```

  - **Définition de la chaîne de connexion**

```
cnx.ConnectionString = cheminBD
```

  - **Connexion à la BD**

```
cnx.Open
```

- **Connexion à la BD avec un DSN**
  - **Déclaration de la variable**

```
Dim cnx As New ADODB.Connection
ou
Dim cnx As ADODB.Connection
Set cnx = New ADODB.Connection
```

  - **Définition de la chaîne de connexion**

```
Const DSN = "BDGSF2012" 'DSN de la BD
cnx.ConnectionString = "DSN=" & DSN & ";"
```

  - **Connexion à la BD**

```
cnx.Open cnxString, userName, passWord, adAsyncConnect
```


VI. Le modèle ADO

- **Exemple :**

```
Const BD = "D:\Cours\BD\Access\Support\BDGSF2024.accdb"
Const DSN = "BDGSF" 'DSN de la BD

Public Function initCon(dsn As String, userName As String, passWord As String,  cnx As ADODB.Connection) As Boolean
 Dim sql As String, cnxString As String, requeteOk As Boolean
 Dim rst As New ADODB.Recordset

 initCon = False
  'initialise la chaine de connexion
 cnx.ConnectionString = "DSN=" & dsn & ";"
  'vérifie que la connexion est bien fermée
 If cnx.State = adStateOpen Then
  cnx.Close
 End If
 On Error GoTo badConnexion
  'connexion a la BD
 cnx.Open cnxString, userName, passWord, adAsyncConnect
 'Attente que la connexion soit établie
  While (cnx.State = adStateConnecting)
  DoEvents
 Wend
```

```
'vérification des erreurs si mauvaise connexion
 If cnx.Errors.Count > 0 Then
  MsgBox cnx.Errors.Item(0)
  initCon = False
  Exit Function
 Else
  initCon = True
 End If
 Exit Function
badConnexion:
 If cnx.Errors.Count > 0 Then
  MsgBox cnx.Errors.Item(0)
  initCon = False
  Exit Function
 Else
  MsgBox Err.Description
 End If
End Function
```

VI. Le modèle ADO

6.5. Exécuter une requête

- **On utilise l'objet Recordset.**
- **Déclaration de la variable**
  - **Dim rst As New ADODB.Recordset**
    ou
  - **Dim rst As ADODB.Recordset**
  - **Set rst = New ADODB.Recordset**

- **Exécution de la requête :**
  - **On utilise la méthode Open de ADODB.Recordset**
  - **Ses paramètres sont :**
    - la requête ;
    - la connexion sur laquelle la requête sera exécuter ;
    - le type de blocage (optionnel) ;
    - le type de requête (optionnel) ;
  - **Sql = "SELECT numCl, nom, adresse FROM Client;"**
  - **rst.Open Sql, cnx**
    - l'ensemble des enregistrements retournés sont dans l'objet Recordset rst.

65

VI. Le modèle ADO

- **Exemple :**

```
Public Function execSql(sql As String, ByRef rst As ADODB.Recordset,
     ByRef cnx As ADODB.Connection) As Boolean
 'initialisation du recordset
 If rst.State <> adStateClosed Then
  rst.Close
 End If
 'ouvre une transaction pour ne pas à avoir de commit en fin de traitement
 cnx.BeginTrans
 rst.CursorLocation = adUseClient 'positionne le curseur coté client
 Set rst.ActiveConnection = cnx 'vérifie que la connexion passée est bonne
 On erro GoTo ErrHandle
 rst.Open sql, cnx 'Exécute la requete
 cnx.CommitTrans 'validation de la transaction
 execSql = True
 Exit Function
ErrHandle:
 execSql = False
 MsgBox "ADOManager.Execsql:ErrHandle" & vbCr & vbCr & Err.Description, vbCritical
End Function
```


66

VI. Le modèle ADO

6.6. Manipulation des données

Accès aux données

Méthode	Description
rst!["nomDuChamp"] ou rst(indice)	Accès à un champ
rst.Fields(indice).Value	Accès à un champ
rst.Fields(indice).Name	Nom d'une colonne
rst.Fields.Count	Nombre de colonnes
Rst.RecordCount	Nombre de lignes

- **Les indices vont de 0 à rst.Fields.Count**
- **Exemples :**
  - **rst.Field(1)** ou **rst.Field("nom")** ou
  - **rst(1)** ou **rst!["nom"]**

Lecture séquentielle des enregistrements

- **Méthode de Recordset pour parcourir le recordset :**

Méthode	Description
MoveNext	Aller à l'enregistrement suivant
MovePrevious	Aller à l'enregistrement précédent
MoveFirst	Aller au premier enregistrement
MoveLast	Aller au dernier enregistrement

VI. Le modèle ADO

6.6. Exemples sans DSN

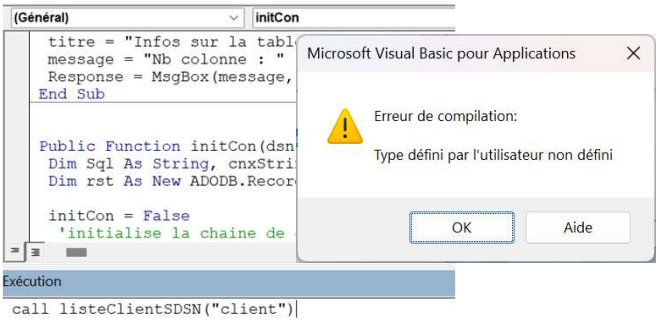
Requête sélection : lister

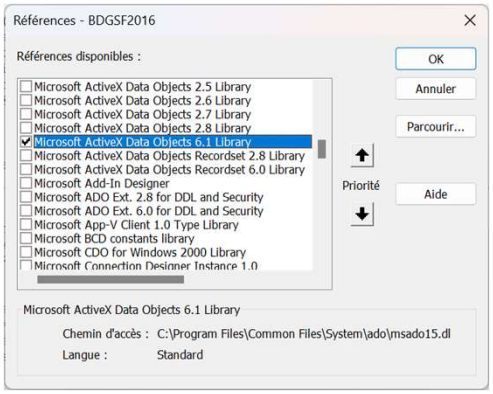
```
Public Sub listeClientSDSN(nomTable As String)
 Dim Sql As String, rst As ADODB.Recordset, cnx As
ADODB.Connection
 Dim res As Boolean, nbEnreg, nbCol As Long
 Dim ch As String, i As Integer, entete As String

 Set rst = New ADODB.Recordset
 Set cnx = New ADODB.Connection
 cnx.Provider = "Microsoft.ACE.OLEDB.16.0"
 cnx.ConnectionString = BD
 cnx.Open 'connexion a la BD
 Sql = "SELECT * FROM " & nomTable & ";"
 res = execSql(Sql, rst, cnx)
 rst.MoveLast 'afin de pouvoir compter les tuples
 nbEnreg = rst.RecordCount 'nb enregistrement
 nbCol = rst.Fields.Count 'nb de colonnes
 For i = 0 To nbCol - 1
   entete = entete & rst.Fields(i).Name & " " 'nom des colonnes
 Next
```

```
 rst.MoveFirst
  Do While Not rst.EOF 'Traitement
   For i = 0 To nbCol - 2
    ch = ch & rst.Fields(i).Value & " " 'ou rst(i)
   Next
   ch = ch & vbCrLf
   rst.MoveNext
  Loop
  Debug.Print entete
  Debug.Print ch
  Debug.Print "Il y a "; nbEnreg; "
enregistrement dans la table " & nomTable
  cnx.Close
End Sub
```

VI. Le modèle ADO

- **Tester la fonction**
  - **call listeClientSDSN("Client")**



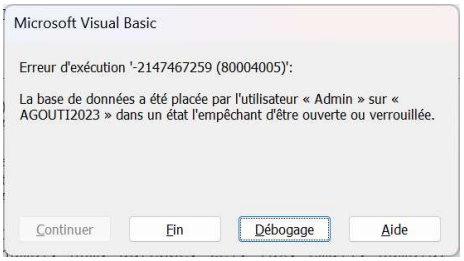


  - **Erreur d'exécution**
  - **Arrêter l'exécution du module**
  - **Ouvrir le Menu Outils/Références**
  - **Cocher la librairie "Microsof ActiveX Data Objects 6.1 Library"**
  - **Valider et recommencer l'exécution**

VI. Le modèle ADO

```
Exécution
call listeClientSDSN("Client")
numCl nom adresse tel fax email numCId
1 Toto Yakro 30642020 30642020 toto@yahoo.fr
2 Frotomougou Abidjan 22343434 22343434
3 Koffi Adjoua Bouaké 30636565 30636565
4 Séri Daloa 34343434 34343434 seri@gmail.com
5 Koffi Yakro 0 0
45 Kouakou Bouaké 31 31 kouakou
145 Kouadio Tipatipa 30 30 kouadio

Il y a  7  enregistrement dans la table Client
```

- **Remarque :**



  - **Si vous avez cette erreur :**
    1. **Cliquer sur Fin**
    2. **Fermer la BD.**
    3. **Ouvrir Access en mode Administrateur**
    4. **Ouvrir la BD**

VI. Le modèle ADO

- **Requête sélection : Rechercher**

```
Public Function rechercherClientSDSN(numC As Long) As Boolean
 Dim Sql As String, rst As ADODB.Recordset, cnx As ADODB.Connection
 Dim res As Boolean
 Set rst = New ADODB.Recordset
 Set cnx = New ADODB.Connection
 cnx.Provider = "Microsoft.ACE.OLEDB.16.0"
 cnx.ConnectionString = BD
 cnx.Open 'connexion a la BD
 Sql = "SELECT * FROM Client WHERE numCl = " & numC & ";"
 res = execSql(Sql, rst, cnx)
 If Not rst.EOF Then 'Traitement
  Debug.Print rst![numCl]; " "; rst![nom]; " "; rst![adresse]
  rechercherClientSDSN = True
 Else
  Debug.Print "Client de numéro "; numC; " non trouvé."
  rechercherClientSDSN = False
 End If
 cnx.Close
End Function
```

- **call rechercherClientSDSN(2)**

Exécution

```
call rechercherClientSDSN(2)
 2  Frotomougou Abidjan
```


71

VI. Le modèle ADO

- **Requête de mise à jour : INSERT**

```
Public Sub ajoutClientSDSN()
 Dim Sql As String, rst As ADODB.Recordset, cnx As ADODB.Connection
 Dim res As Boolean
 Set rst = New ADODB.Recordset
 Set cnx = New ADODB.Connection
 cnx.Provider = "Microsoft.ACE.OLEDB.16.0"
 cnx.ConnectionString = BD
 cnx.Open 'connexion a la BD
 Sql = "INSERT INTO client(numCl, nom, adresse, tel, fax, email) "
 Sql = Sql & "VALUES(2956,'qwertz', 'Abj', 30645555, 30645555,
'qwertyz@hotmail.com', 'C008')"
 res = execSql(Sql, rst, cnx) 'execution de la requete
 cnx.Close
End Sub
```

- **call ajoutClientSDSN**
- **call listeClientSDSN("client")**

Exécution

```
call listeClientSDSN("client")
numCl nom adresse tel fax email numCId
1 Toto Yakro 30642020 30642020 toto@yahoo.fr
2 Frotomougou Abidjan 22343434 22343434
3 Koffi Adjoua Bouaké 30636565 30636565
4 Séri Daloa 34343434 34343434 seri@gmail.com
5 Koffi Yakro 0 0
45 Kouakou Bouaké 31 31 kouakou
145 Kouadio Tipatipa 30 30 kouadio
2956 qwertz Abj 30645555 30645555 qwertyz@hotmail.com
```

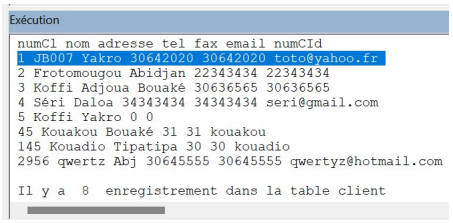

72

VI. Le modèle ADO

- **Requête de mise à jour : UPDATE**

```
Public Sub modifierClientSDSN(numC As Long, nomCl As String)
 Dim Sql As String, rst As ADODB.Recordset, cnx As ADODB.Connection
 Dim res As Boolean
 Set rst = New ADODB.Recordset
 Set cnx = New ADODB.Connection
 cnx.Provider = "Microsoft.ACE.OLEDB.16.0"
 cnx.ConnectionString = BD
 cnx.Open 'connexion a la BD
 Sql = "UPDATE Client SET nom = '" & nomCl & "' WHERE numCl = " & numC
 res = execSql(Sql, rst, cnx) 'execution de la requete
 cnx.Close
End Sub
```

- **call modifierClientSDSN(1, "JB007")**
- **call listeClientSDSN("client")**

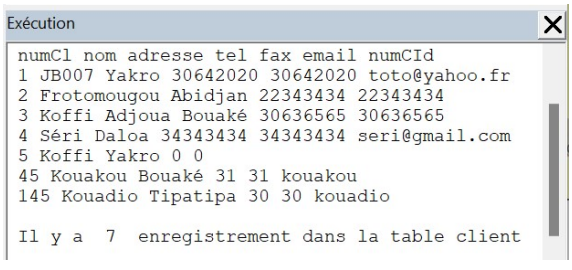


VI. Le modèle ADO

- **Requête de mise à jour : DELETE**

```
Public Sub supprimerClientSDSN(numC As Long)
 Dim Sql As String, rst As ADODB.Recordset, cnx As ADODB.Connection
 Dim res As Boolean
 Set rst = New ADODB.Recordset
 Set cnx = New ADODB.Connection
 cnx.Provider = "Microsoft.ACE.OLEDB.16.0"
 cnx.ConnectionString = BD
 cnx.Open 'connexion a la BD
 Sql = "DELETE FROM Client WHERE numCl = " & numC
 res = execSql(Sql, rst, cnx) 'execution de la requete
 cnx.Close
End Sub
```

- **call supprimerClientSDSN (2956)**
- **call listeClientSDSN("client")**



VI. Le modèle ADO

- **Exercice 1 :**
  - **Créer une fonction qui retourne la quantité en stock d'un produit.**
- **Exercice 2 :**
  - **Créer une procédure de mise à jour de la quantité en stock d'un produit.**
- **Exercice 3 :**
  - **Ajouter des boutons de navigation dans le formulaire Client :**
    - Premier (premier enregistrement);
    - Suivant (dernier enregistrement);
    - Précédent (premier enregistrement) ;
    - Dernier (dernier enregistrement);
    - Ajout (Ajout d'un nouvel enregistrement);
    - Suppression (Suppression d'un enregistrement);
    - Modification (Modification d'un enregistrement).

VI. Le modèle ADO

- **6.7. Exemples de code avec DSN**
  - **Requête sélection : Liste des clients**

```
Public Sub listeClientADODSN()
 Dim sql As String, rst As ADODB.Recordset, cnx As ADODB.Connection
 Dim res, D As Boolean
 Set rst = New ADODB.Recordset
 Set cnx = New ADODB.Connection
 'connexion a la BD
 D = initCon("bdgsf", "Totoe", "TotoAli", cnx)
 sql = "SELECT * FROM Client ;"
 'execution de la requete
 res = execSql(sql, rst, cnx)
 'Traitement
 rst.MoveLast 'afin de pouvoir compter les tuples
 nbClient = rst.RecordCount
 Debug.Print "Il y a "; nbClient; " enregistrement dans la table client "
 rst.MoveFirst
 Do While Not rst.EOF 'Traitement
  Debug.Print rst![numCl]; " "; rst![nom]; " "; rst![adresse]
  rst.MoveNext
 Loop
 cnx.Close
End Sub
```

Bibliographie

- **Livre**
  - "Modélisation dans la conception des systèmes d'information", Edition Masson.
  - "Les fichiers et organisation des données", C. JOUFFROY, C.LEITANG, Bordas Informatique.
  - "Base de données et systèmes relationnels", C. DELOBEL, M. ADIBA, Dunod Informatique.
  - "Les bases de données relationnelles", Serge MIRANDA, José Maria BUSTA.
  - "Base de données, les systèmes et leurs langages", G. GARDARIN, Edition Eyrolles.
  - "Système d'information et base de données", GALACSI, Bordas Informatique.

Bibliographie

- **Webogaphie**
  - http://www.infres.enst.fr/~dombd/polyv7/
  - http://lbdwww.epfl.ch/f/teaching/courses/poly2/11/11.htm
  - Cours de Yolaine.Bourda@supelec.fr
  - Cours de Mme Silber cours@www-aius.u-strasbg.fr