



# Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour

EA 3000

## R3.07 - SQL et programmation

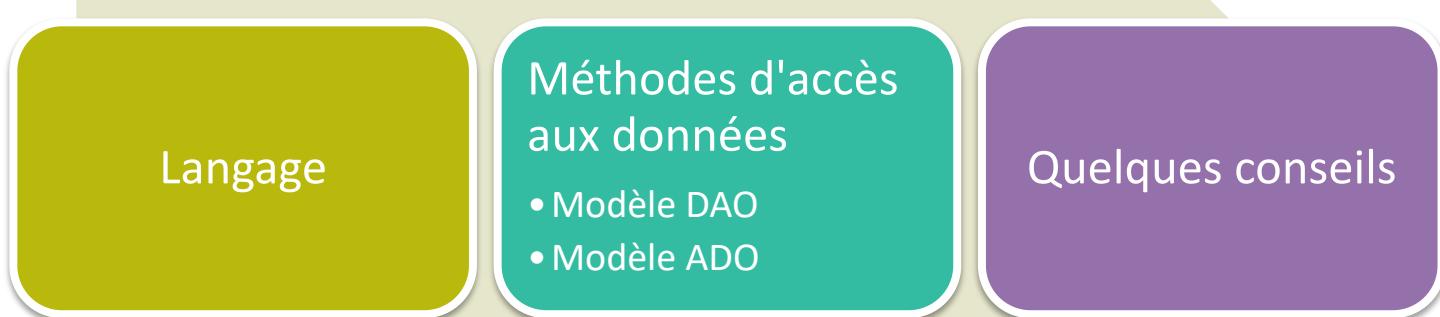
Rappel

DAO/ADO

PLSQL

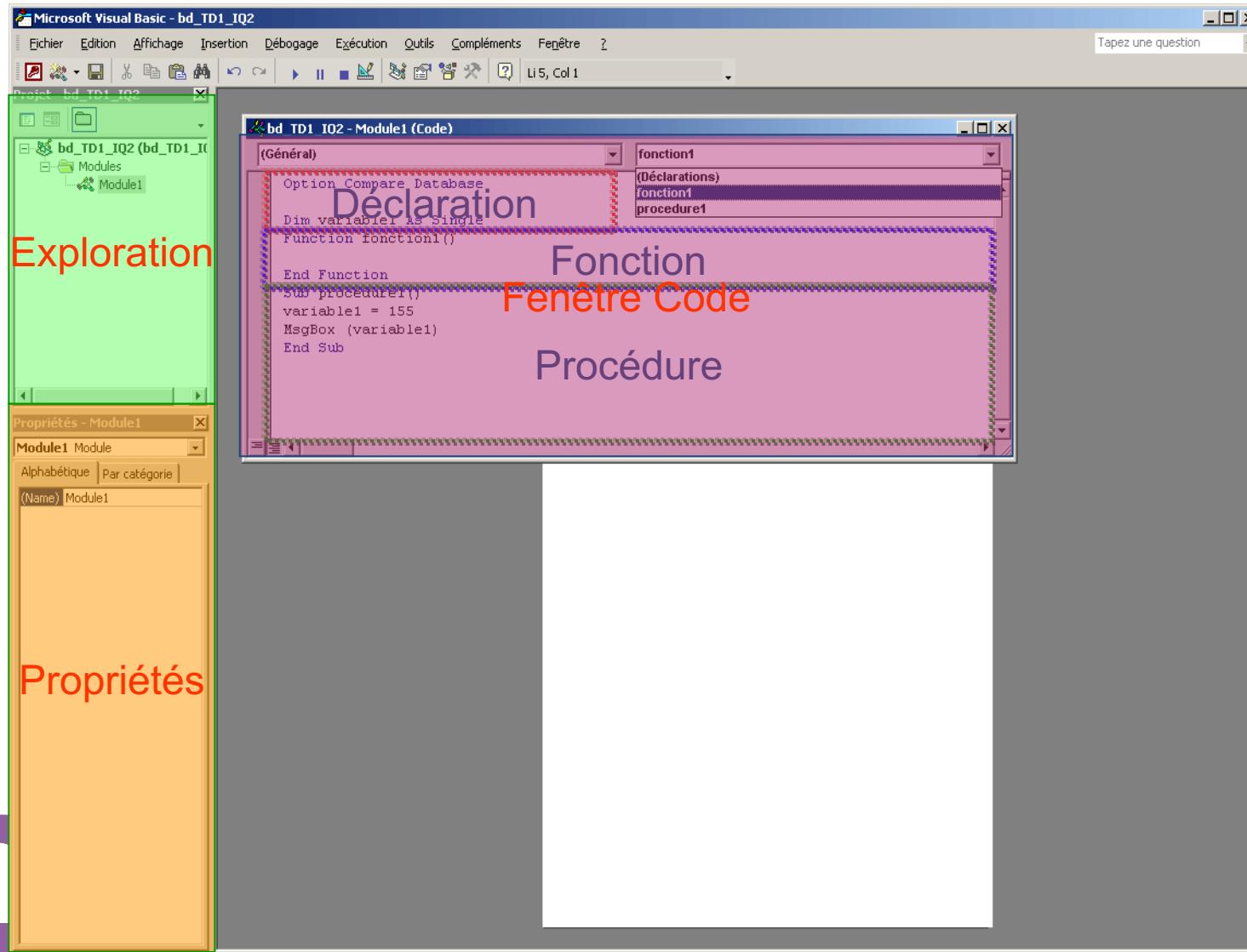


# Plan



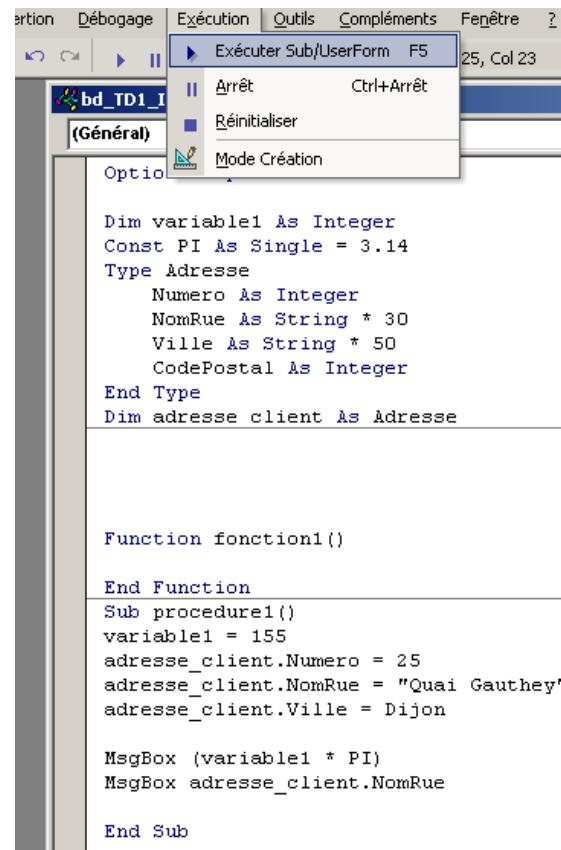
# Environnement de développement IDE

## Microsoft Access



# Comment tester ?

## Exécution d'une procédure



The screenshot shows the Microsoft Visual Basic Editor (VBE) interface. The menu bar is visible at the top, with 'Exécution' (Execution) being the active tab. A dropdown menu from this tab is open, showing options: 'Exécuter Sub/UserForm F5' (Run Sub/UserForm F5), 'Arrêt Ctrl+Arrêt' (Break Ctrl+Break), 'Réinitialiser' (Reset), and 'Mode Création' (Create mode). The main code window displays the following VBScript code:

```
Dim variable1 As Integer
Const PI As Single = 3.14
Type Adresse
    Numero As Integer
    NomRue As String * 30
    Ville As String * 50
    CodePostal As Integer
End Type
Dim adresse_client As Adresse

Function fonction1()

End Function
Sub procedure1()
variable1 = 155
adresse_client.Numero = 25
adresse_client.NomRue = "Quai Gauthey"
adresse_client.Ville = Dijon

MsgBox (variable1 * PI)
MsgBox adresse_client.NomRue

End Sub
```

# Composants Microsoft Access



# Composants majeurs

## ■ Access comporte deux composants :

### ■ Microsoft Jet

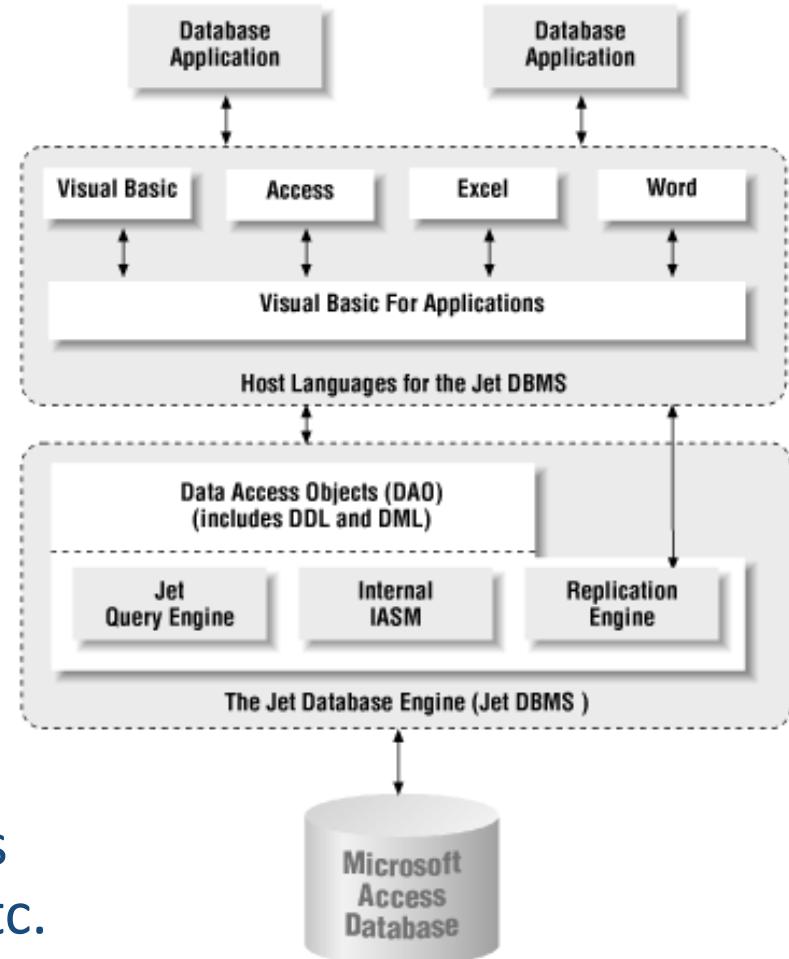
- Contrôle le stockage des données
- Définit les objets de la BD

### ■ Le moteur de l'application

- Contrôle la programmation
- Contrôle l' interface

## ■ Comment ça fonctionne ?

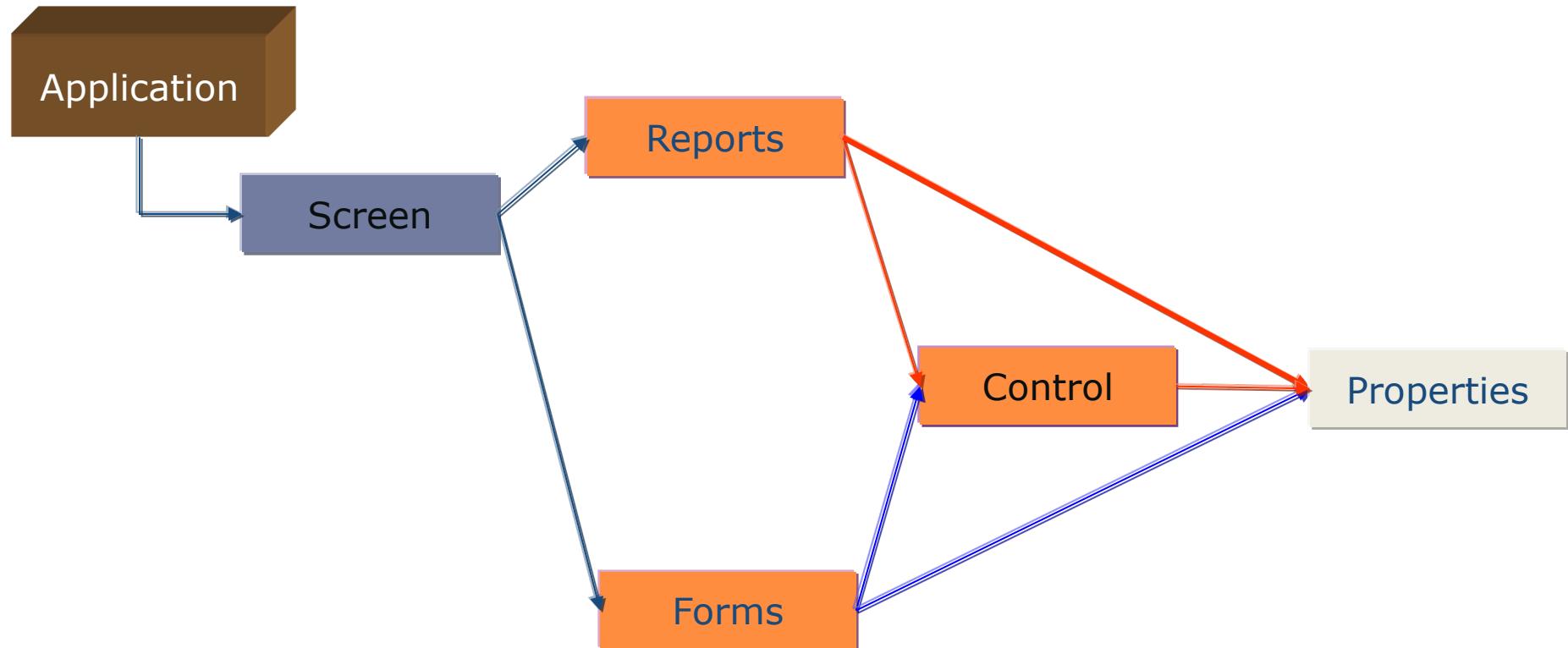
- Quand vous ouvrez une BD, le moteur de l'application utilise Microsoft Jet pour déterminer les noms des tables, des requêtes, etc.



- C'est le Gestionnaire de fichiers
- Il possède les caractéristiques suivantes :
  - Moteur à 64 bits
  - Support Unicode
    - Jeux de caractères à deux octets par caractère
  - Types de données compatibles avec SQL Server
  - SQL 92
  - Verrouillage amélioré des données



# Structure simplifiée du modèle Application



# Le langage VBA



# Types de données

## ■ Boolean

- True et False

## ■ Integer (ou %)

- Entiers entre -32 768 et 32 767

## ■ String (ou \$)

- Entre 0 et environ 63 Ko de caractères

## ■ Decimal

## ■ Date

- Pour stocker les dates (**1/01/100 – 31/12/9999**) et les heures

## ■ Variant (Par défaut)

- Type de données particulier pouvant contenir des données numériques, des chaînes ou des dates, des types définis par l'utilisateur ainsi que les valeurs spéciales **Empty** et **Null**



# Types de données



# Déclarations

## ■ Des variables (**DIM**)

- DIM entier as INTEGER
- DIM type as Variant
- DIM x(10, 25) as Single

## ■ Des constantes (**CONST**)

- Const PI As Single = 3.14



# Déclarations

## ■ Des procédures (SUB)

```
Sub Attribution_Note_Aleatoire()
Dim Notes As Recordset

    . . .

End Sub
```

## ■ Des fonctions (function)

```
Function NoteAleatoire() As Single
Dim ValeurAlea As Single
    ValeurAlea = Rnd() * 20
    NoteAleatoire = Format(ValeurAlea, "0.0")
End Function
```



# Portée d'une variable

## ■ Private

- Permet de définir des variables privées
- Une procédure privée englobe des variables privées

## ■ Public (ou Global)

- Permet de définir des variables publiques

## ■ Static

- Permet de (re)définir des variables dont le contenu est non modifiable



# Branchement et boucles

## ■ Instruction IF

IF (a=5) Then

...

Else

...

ENDIF

## ■ Instruction If(*condition, truepart, falsepart*)

## ■ Instruction GOTO

GoTO Fin

.....

:FIN

## ■ Instruction FOR...NEXT

For i=1 to 10

---

Next i

## ■ Instruction While

While i<=10

---

Wend



# Branchement et boucles

## ■ Instruction DO..LOOP

- **Do** [{While | Until} condition]  
[statements]

**Loop**

- Vous pouvez également utiliser la syntaxe suivante :

- **Do**  
[statements]  
**Loop** [{While | Until} condition]

## ■ Instruction Select Case

Select CASE valeur

Case 0 to 2

....

Case 3 to 5

...

Case 6

Case ELSE

END SELECT

## ■ L' instruction CALL

- Transfère le contrôle à une procédure ou à une Fonction (interne ou externe)
- Call MyProc(0)



# Référencer des objets

■ Plusieurs méthodes, mais principalement

■ **NomCollection("Nom de l'Objet")**

■ **Ex: Forms("Clients")**



# Gestion d'erreurs

## ■ On error ... Génération d'une constante Err

## ■ Resume NEXT

- Le programme continue sans abandonner

## ■ ERL

- Renvoie le numéro de la ligne où l'erreur s'est produite

## ■ Err.Number

- Contient le code de l'erreur

## ■ Error\$(Err) ou Err.Description

- Donne les détails de ERR

```
On Error GoTo Err_Click
...
Err_Click:
MsgBox Err.Number & " : " & Err.Description
```



# Fonctions intégrées

## ■ Sur les chaînes

- Left (chaîne, taille), Right, Replace, etc.

## ■ Arithmétiques

- ABS, LOG, EXP, etc.

## ■ Commandes

- CHDIR, CHDRIVE, DIR\$, MKDIR, RMDIR, etc.

## ■ Heure/date

- Date\$, Now, etc.

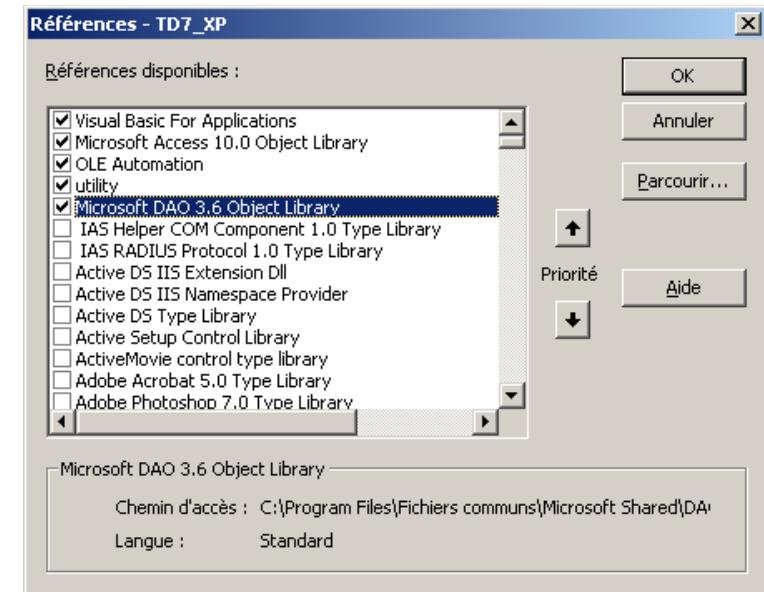
## ■ Affichage

- MSGBOX, INPUTBOX\$, etc.



# Librairies

- Les librairies proposées dans très variées
- Pour les intégrer dans un module:
- Dans l'IDE de Visual Basic
  - Outils/références



- Intégrer uniquement les librairies concernées, sinon 😞



# Méthodes d'accès aux données



# Méthodes d'accès aux données

## ■ Plusieurs librairies sont proposées. Mais Principalement :

### ■ DAO (Data Access Objects)

- Interface permettant l'accès aux données qui communique avec Microsoft Jet et des sources de données compatibles ODBC pour se connecter à, récupérer, manipuler et mettre à jour des données et la structure de base de données.
- On fait du SQL ACCESS (\*, ?, etc.)

### ■ ADO (ActiveX Data Objects)

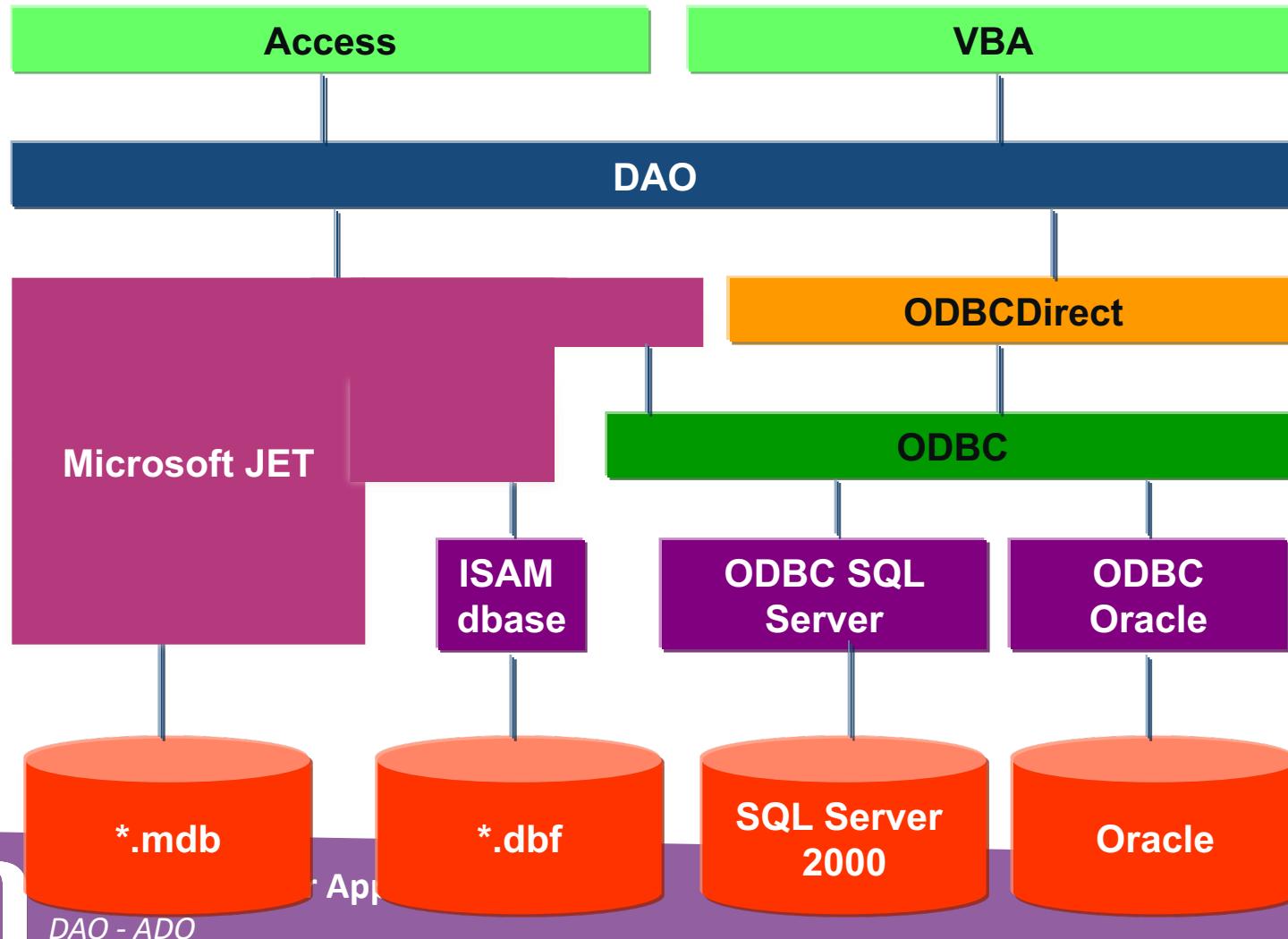
- Interface d'accès aux données qui communique avec des sources de données compatibles OLE DB pour la connexion, la récupération, la manipulation et la mise à jour de données (Via le Web par exemple ;)
- On fait du SQL Standard (%, \_, etc.)



# Le modèle DAO



# Le modèle DAO

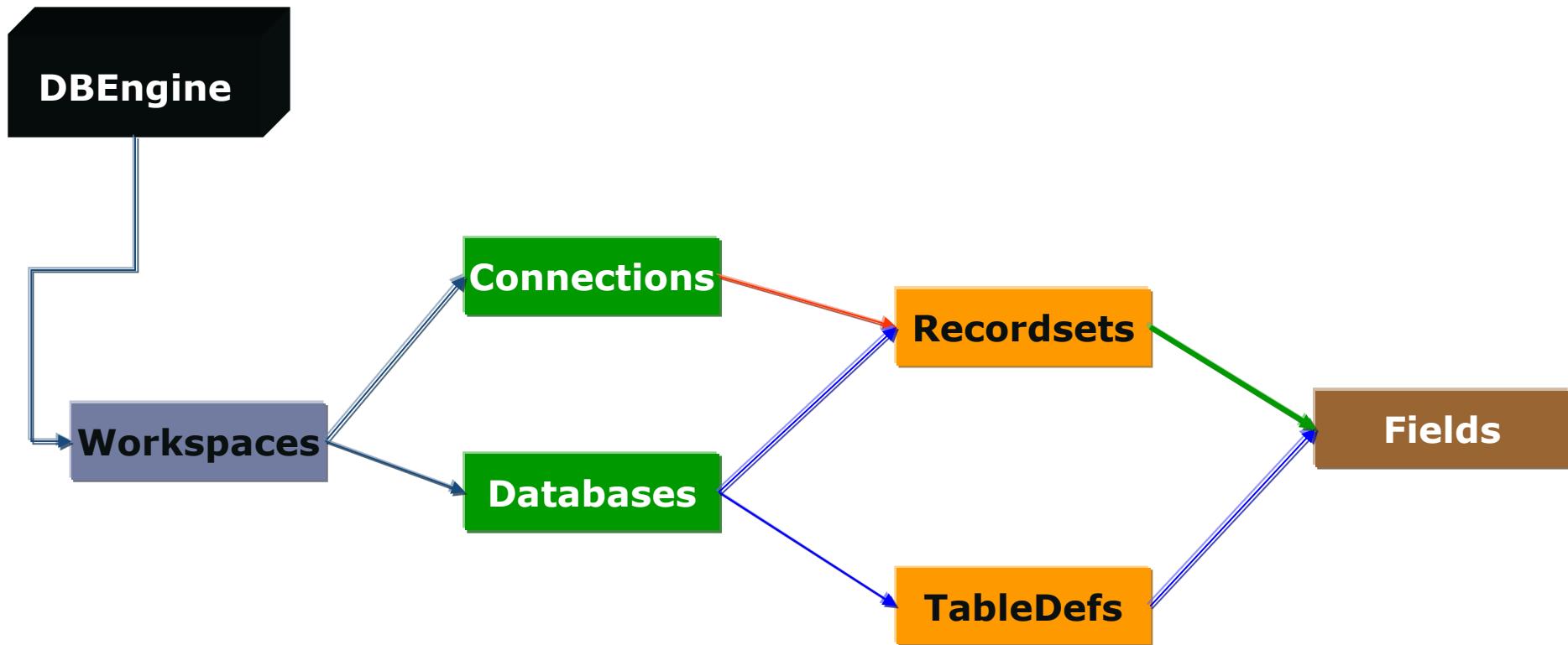


# Caractéristiques de DAO

- Adapté à la gestion des BD hétérogènes
  - Oracle, SQL server, Access, Sybase, Paradox, etc.
- Conçu pour des applications client/serveur
- Dépend d'un groupe international (et pas de Microsoft ☺)



# Structure du modèle DAO



# Accès à une BD

## ■ Ouverture

- OpenDataBase(Name [, Exclusif[, ReadOnly]])

## ■ Fermeture

- Close

## ■ Exemple (Déclaration de trois bases)

```
Sub Accès_femeture_base()
    ' 1- Déclaration des bases
    DIM MaBD1 As DAO.DataBase
    DIM MaBD2, MaBD3 As DataBase
    ' ...
    ' 2- Affectation des bases
    Set MaBD1 = OpenDataBase("c:\temp\fichierDB1.mdb")      ' base locale
    Set MaBD2 = OpenDataBase("\\serveur_IUT\Partage\fichierDB2.mdb", TRUE) ' base partagée
    en mode exclusif
    Set MaBD3 = currentDb()
    ' ...
    ' 3- Fermer les bases
    MaBD1.Close
    MaBD2.Close
    MaBD3.Close
End Sub
```



# Tables/Requêtes

## Déclaration

```
Sub Attribution_table()
    '1- déclaration de la base
    Dim db As Database

    '2- déclaration recordset
    Dim definition_table As DAO.Recordset
    Dim requete As DAO.Recordset

    '3- Affectation
    Set db = CurrentDb()
    Set definition_table = db.OpenRecordset("Etudiants", dbOpenDynaset)
    Set requete = db.OpenRecordset("Requête Clients lyonnais", dbOpenDynaset)

End Sub
```



# Accéder aux enregistrement d'une table

- Plusieurs moyens, mais principalement :

## RecordSet



# RecordSet

## ■ Applicable sur les tables et les requêtes

## ■ Trois éléments sont essentiels

- La base de données concernée
- Les enregistrements dans la base
- Le type de RecordSet

## ■ Déclaration d'une variable

### ■ Pour représenter la table "Clients"

```
'1- déclaration des variables
Dim db As DAO.Database
Dim tb_clients As DAO.Recordset
'2- Affectation
Set db = CurrentDb()
Set tb_client = db.OpenRecordset("Clients", dbOpenDynaset)
```

DbOpentable)

dbOpensnapshot)



# RecordSet

## ■ Propose plusieurs méthodes :

### ■ De positionnement

- MoveFirst | MoveNext | MoveLast | MovePrevious | Move n
- BOF, EOF

### ■ De recherche

- FindFirst | FindLast | FindNext | FindPrevious

### ■ De manipulation

- Delete
- Update
- Edit
- Addnew
- Field



# RecordSet

## ■ Exemple

- On veut savoir si on a des clients dont le code commence par A

```
'1- déclaration des variables
Dim db As Database
Dim tb_client As Recordset
'2- Affectation
Set db = CurrentDb()
Set tb_client = db.OpenRecordset("SELECT * FROM Clients where
[Code Client] LIKE 'A*' ", dbOpenDynaset)
'3- Rechercher les clients dont le code commence par A
If Not tb_client.EOF Then
    MsgBox "trouvé"
End If
```



# Comment accéder à un champ ?

## • En utilisant

### ■ Fields :

- stocke les différents champs d'un enregistrement. Chaque champ est représenté par un objet instance de la classe Field
- Le nombre de Fields est déterminé par la méthode [Count](#)
- Remarque : la classe field permet de représenter un champ. On y trouve principalement les propriétés **name** et **value** qui renvoie respectivement le nom et la valeur du champ.

### ■ Exemple

```
Dim rs As Recordset
rs ("nom_produit")                                ' valeur du champ nom_produit
```



# Ajout d'un enregistrement

## ■ Elle se fait en respectant les étapes suivantes :

1. Utiliser la méthode **AddNew** du Recordset pour créer un nouvel enregistrement vide et s'y positionner
2. Donner une valeur aux champs (**Fields**) du Recordset
  - On peut également passer ces valeurs comme paramètres de la méthode AddNew
3. Utiliser la méthode **Update** du Recordset pour enregistrer le nouvel enregistrement dans la base
  - On peut utiliser la méthode **CancelUpdate** pour annuler la création



# RecordSet

## ■ Exemple d'ajout

On veut ajouter le client Info2 ayant le code 'IUT'

```
Dim db As Database
Dim tb_client As Recordset

Set db = CurrentDb()
Set tb_client = db.OpenRecordset("Clients", dbOpenDynaset)

' 1- Demander la création
Tb_client.AddNew

' 2- Fournir les valeurs des champs
Tb_client("Code Client") = "IUT"
Tb_client("Nom") = "Info2"

' 3- Enregistrer les données
Tb_client.Update
```



# RecordSet

## Modification d'enregistrements

1. Se positionner sur l'enregistrement à modifier
2. Utiliser la méthode Edit
3. Modifier la valeur des champs
4. Utiliser la méthode update

## Exemple

On veut changer le nom du client dont le code est 'ANTON'

```
Dim db As Database
Dim tb_client As Recordset
Set db = CurrentDb()
Set tb_client = db.OpenRecordset("Select * from Clients where [Code client] = 'ANTON' ")

If Not tb_client.EOF Then
    MsgBox "trouvé"
    tb_client.Edit
    tb_client("Nom") = "Dupond"
    tb_client.Update
End If
```



# RecordSet

## ■ Suppression d'enregistrements

1. Se positionner sur l'enregistrement à supprimer
2. Utiliser la méthode delete

Remarque : l'enregistrement courant n'est plus valide ... pensez donc à le déplacer (avec MoveNext ou autre)

## ■ Exemple

### ■ Supprimer le client dont le Code est ANTON

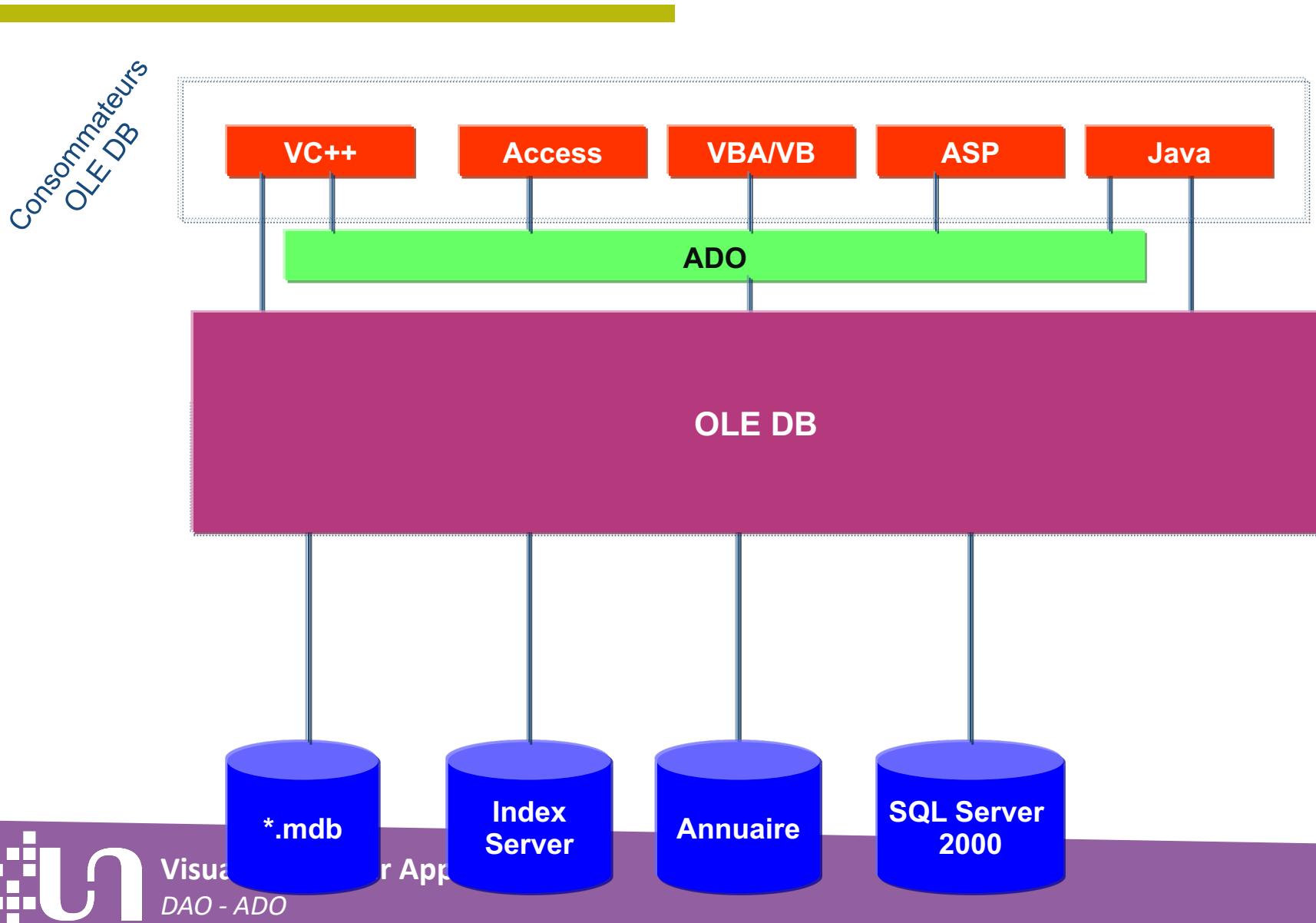
```
Dim db As Database
Dim tb_clients As Recordset
Set db = CurrentDb()
Set tb_client = db.OpenRecordset("...", dbOpenDynaset)
tb_client.delete
tb_client.MoveNext
```



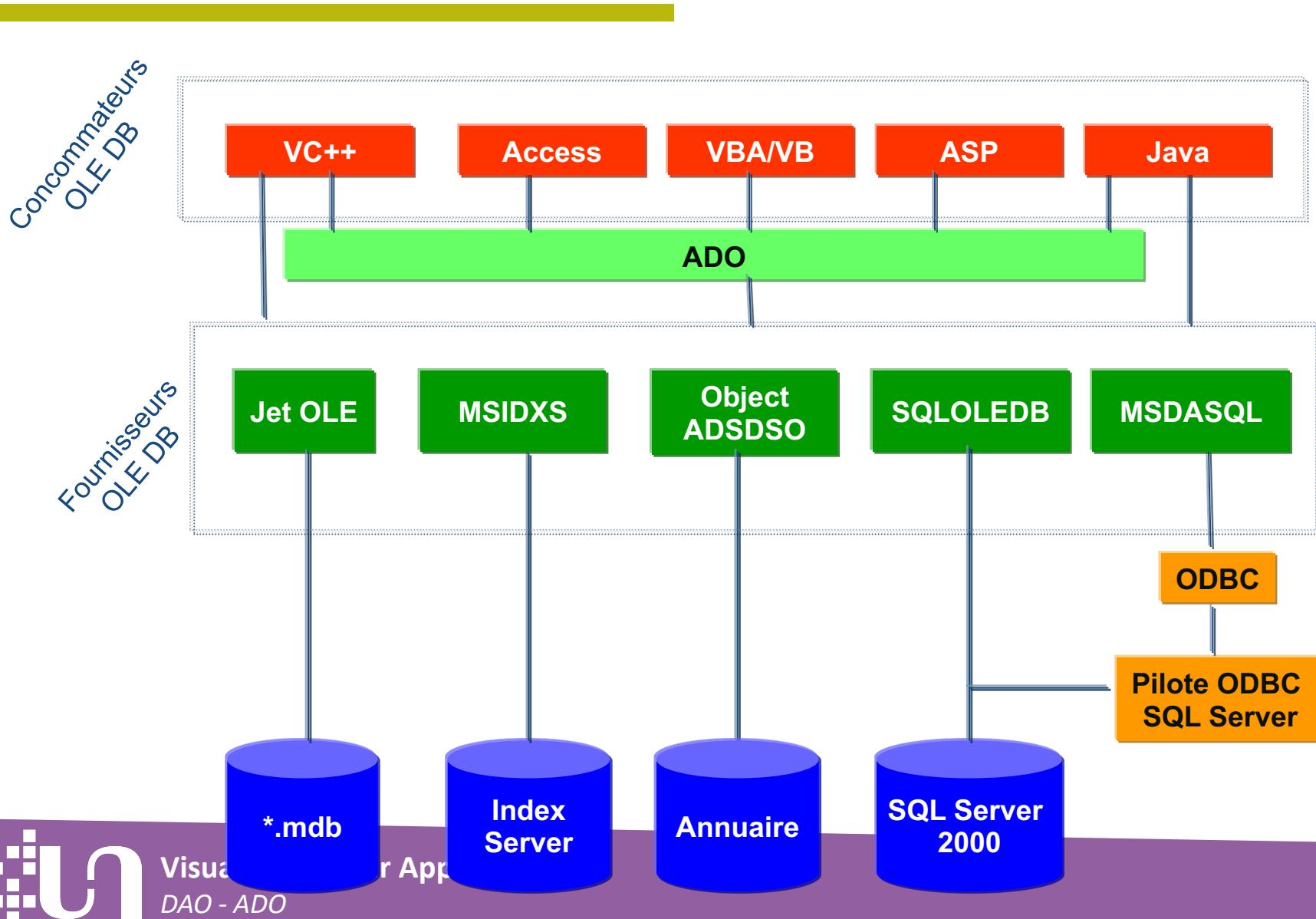
# Le modèle ADO



# Le modèle ADO



# Le modèle ADO



# Fournisseurs OLE DB

- **Jet OLE DB 4.0**
  - Pour les BD Access
- **SQL Server**
- **Oracle**
- **ODBC Drivers**
  - Pour les sources de données ODBC
- **OLAP Services**
  - Pour le serveur OLAP Microsoft (Exchange Server)
- **Simple Provider**
  - Pour les fichiers de texte simples
- **Microsoft Directory Services**
  - Pour Active Directory sous Windows 2000
- **Internet Publishing**
  - Pour l'accès aux serveurs Web
- **DTS packages**
  - Pour les services de conversion de données SQL server



# Caractéristiques de OLE DB et ADO

## ■ Adaptés à la gestion de toute *source de données*

- BD, systèmes de messagerie, service d'annuaires, serveurs Web, etc.

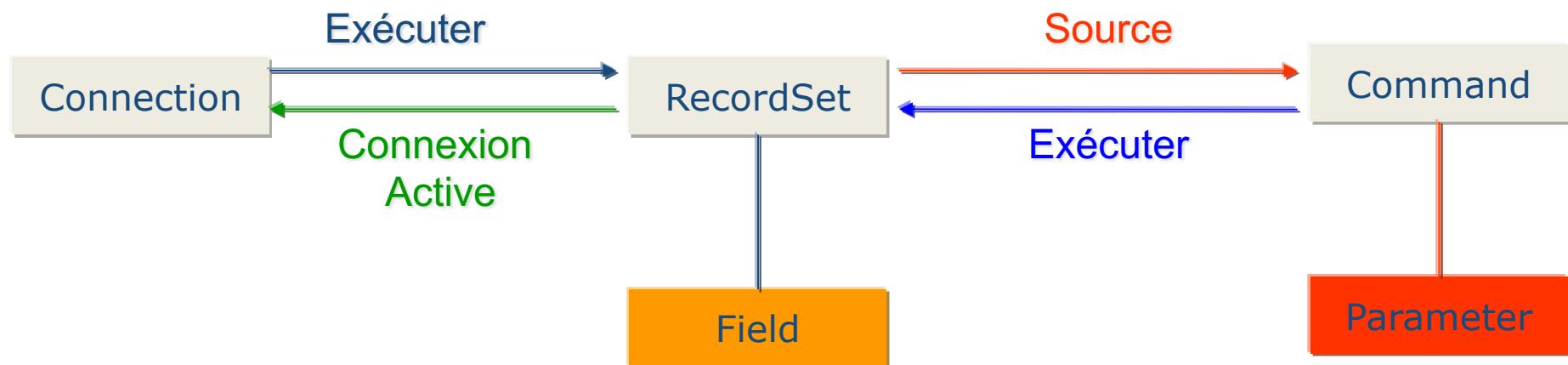
## ■ Conçus pour des applications orientées Internet

- Jeux d'enregistrements déconnectés, meilleure gestion des utilisateurs, etc.

## ■ Dépend complètement de Microsoft ☺



# Structure simplifiée



# Instanciation des classes

## ■ Syntaxe

```
Dim nom_variable As New ADODB.nom_classe
```

## ■ Exemple

```
Dim cn As New ADODB.Connection  
Dim cmd As New ADODB.Command  
Dim rst As New ADODB.Recordset
```



# La classe Connection

- Permet la connexion à une source de données
- Un objet de cette classe identifie une et une seule connexion à une source de données
- Permet l'exécution de commandes
  - Requête de mise à jour, d'insertion, de suppression, etc.



# Les méthodes Connection

## ■ Open

- Ouvre une connexion à une source de données

```
Dim cn As New ADODB.Connection
Set cn = CurrentProject.Connection

' cn.open "Provider=Microsoft.Jet.OLEDB.4.0; Data source = c:\temp\Ma_Base.mdb"
```

## ■ Execute

- Exécute une requête et récupère le résultat dans un RecordSet

```
Dim rs1, rs2 As New ADODB.Recordset
Set rs1 = cn.execute("select [nom Client] from Clients")
cn.Execute("insert into ma_table values (5, 'abc', ...)")
```

## ■ Close

- Ferme une connexion

```
cn.close
Set cn = nothing
```



# La classe Command

- Permet d'exécuter des commandes sur une source de données (des instructions SQL)
- Souvent avec des paramètres



# La classe Command

## ■ Propriétés

- .CommandText : stocke le texte de la commande
  - Ex : `objet_cmd.CommandText = "select * from Clients"`
- .ActiveConnection : permet de choisir la connexion (donc la base) sur laquelle s'exécutera la commande
  - Ex : `set objet_cmd.ActiveConnection = cn`



# La classe Command

## ■ Méthode

■ Execute : permet d'exécuter la commande

■ Sans paramètre (stockée dans la propriété CommandText)

■ Ex : *objet\_command.Execute*

■ Avec paramètre

■ Ex : *objet\_command.Execute("select \* from produits")*



# La classe Command

## Exemple

```
Dim cn As New ADODB.Connection
cn.open "DSN=Base_Oracle"
' Cela signifie qu'une source de données nommée
Base_Oracle existe déjà dans ODBC

Dim cmd As New ADODB.Command
cmd.CommandText = "select * from Clients"
Set cmd.ActiveConnection = cn

Dim rs As New ADODB.recordset
Set rs = cmd.Execute
' Set rs = cmd.Execute ("select * from Clients")
```



# La classe Recordset

- Permet de contenir l'ensemble des données extraites des sources
- Stocke le résultat de l' exécution d' une commande sous forme d' un ensemble de lignes
- Seule la ligne courante est visible (Notion de curseur)



# La classe Recordset

## ■ Trois possibilités d'instanciation via la méthode

### ■ Execute de Connection

```
Dim rs As New ADODB.recordset
set rs = cn.execute("select [nom Client] from Clients")
```

### ■ Execute de Command

```
Dim cmd As New ADODB.Command
cmd.CommandText = "select * from Produits"
set cmd.ActiveConnection = cn
Dim rs As New ADODB.recordset
set rs = cmd.Execute
```

### ■ Open de Recordset

```
Dim rs As New ADODB.recordset
rs.open "select * from Clients", "Base_Oracle"
```



# Création d'un Recordset

## ■ Syntaxe générale de la méthode open

```
objet_recordset.open une_instruction connection_active lock_type
```

requête sql, nom d' une table, ...

une chaîne contenant le DSN  
ou une référence à un objet  
de la classe connection

Une constante qui prend une des valeurs suivantes :

- **adLockReadOnly** les données ne peuvent pas être modifiées
- **adLockPessimistic** les enregistrements sont verrouillés dès le début
- **adLockOptimistic** les enregistrements ne sont verrouillés qu'au moment de l'appel de la méthode update



# Manipulation d'un Recordset

## ■ Propose plusieurs méthodes :

### ■ De positionnement

- MoveFirst | MoveNext | MoveLast | MovePrevious | Move n, start
- BOF, EOF

### ■ De recherche

- Find

### ■ De manipulation

- Delete
- Update
- Addnew



# DAO vs ADO



# Ouvrir la base courante

## ■ DAO

```
Dim db As DAO.Database  
Set db = CurrentDb()
```

## ■ ADO

```
Dim objConn As New ADODB.Connection  
Set ObjConn = CurrentProject.Connection
```



# Ouvrir un jeu d'enregistrements

## ■ DAO

```
Dim db As DAO.Database
Dim rs As DAO.RecordSet
Set db = opendatabase("c:\temp\Ma_Base.mdb")
Set rs = db.OpenRecordSet("Select * From Client", dbOpenDynaset)
```

## ■ ADO

```
Dim objConn As New ADODB.Connection
ObjConn.Provider = "Microsoft.Jet.OLEDB.4.0"
ObjConn.Open "Data source = c:\temp\Ma_Base.mdb"
Dim objrs As New ADODB.RecordSet
Objrs.Open "Select * From Client", objConn, adOpenKeySet, adLockOptimistic
```



# Modifier un jeu d'enregistrements

## ■ DAO

```
Dim db As DAO.Database
Dim rs As DAO.RecordSet
Set db = opendatabase("c:\temp\Ma_Base.mdb")
Set rs = db.OpenRecordSet("Select * From Client", dbOpenDynaset)
rs.EDIT
rs("Nom") = "IQ2"
rs.Update
```

## ■ ADO

```
Dim objConn As New ADODB.Connection
ObjConn.Provider = "Microsoft.Jet.OLEDB.4.0"
ObjConn.Open "Data source = c:\temp\Ma_Base.mdb"
Dim objrs As New ADODB.RecordSet
Objrs.Open "Select * From Client", objConn, adOpenKeySet, adLockOptimistic
Objrs("Nom") = "IQ2"
Objrs.Update
```



# Conseils

## ■ Optimiser votre application

- Réduire la consommation de mémoire
  - Utilisez le bon type de données
  - Regroupez les procédures dans les modules
  - Ne chargez pas les bibliothèques inutiles
- Augmenter la vitesse d'exécution
  - Réduire les portions code
  - Utilisez les constantes
- Accroître la vitesse perçue
  - Pré-charger et masquer des formulaires
  - Stocker localement les données dans un cache
- Compackter la base
- Commentez votre code

