# SIntroduction to Databases with VB.net and MS Access

### Aim

* Experience how data can be stored in a database.
* Be able to Create, and alter the structure of the database using DDL
* Be able to Add, Edit, and retrieve data using SQL
* Perform the above operations if VB.Net (or the language of your choice)
* Be able to visualise the structure of the data

## Tasks

You are going to create a database using VB.net and DDL (Data Definition Language)… We are going to use a MS Access File. This is purely because this is easy to view what has happened in college (Using MS access) or at home (Using OpenOffice). We expect you to use a DataBase Server (MySQL) for your computing project. The database you will create will hold some Student Information, Exam information and Exam results.

### Task1 – Create VB Form Project (See end of socument for working with VS 2019)

1. Create a new Forms Project. For ease you can create a new button for each of the following tasks. So by the end you will have a VBform with lots of buttons on it…
2. Name the buttons something sensible (e.g. btnCreateDataBase).. You should also alter the display text so each button can be identified when the program runs.
3. At the start of the program file you need to add the following references  
    (**Project**: **add reference** :**COM**: ADO Ext 2.7 for DDL and Security (ADOX)
4. Now type the following to make use of these libraries

Imports ADOX

Imports System.Data.OleDb

Imports System.IO

1. We’re going to be a smidge lazy and have a few globals that we will use throughout. Put these at the top of the class.

Dim mycommand As OleDbCommand

Dim myconnection As OleDbConnection

Dim myReader As OleDbDataReader

Dim DDLstr As String

Dim SQLstr As String

### Task 2 - Create a Database

I’ll give you the code for creating the database… It will Create a new database file called Test.mdb in the debug bin folder.

Sub CreateDataBase()

'This isn't DDL at all really... But is needed if we are using an MDB file

Dim cat As Catalog = New Catalog()

Try

cat.Create("Provider=Microsoft.Jet.OLEDB.4.0;" & \_

"Data Source=Test.mdb;" & \_

"Jet OLEDB:Engine Type=5")

Msgbox("Database Created Successfully")

Catch ex As Exception

Msgbox("Database Creation failed " & ex.Message)

End Try

cat = Nothing

End Sub

Make your first button call the above procedure on the OnClick event.

### Task 3 Opening and Closing the connection to the Database

From now on before you execute the code for a following tasks the database connection needs to be opened… and after needs to be closed.

Sub OpenDataBase()

Try

myconnection = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Test.mdb")

myconnection.Open()

Msgbox("Database Opened Successfully")

Catch ex As Exception

Msgbox("Database not opened " & ex.Message)

End Try

End Sub

Ensure this command **myconnection.Close()** is ALWAYS executed after connecting to the database

The Button.Click event procedure will look like this

Private Sub cmdOpen\_Click(sender As System.Object, e As System.EventArgs) Handles cmdOpen.Click

OpenDataBase()

<insert future code between open and close

myconnection.Close()

End Sub

### Task 4 – Creating tables

Now we are using DDL proper! We will be creating 3 tables: Students, Exams, Results I’ll give you the code for students and specify the other 2 tables…

Sub CreateStudentTable()

'this is the actual DDL command string we are sending to the database engine. The exact synatx (brackets, quaotes etc..) varies for each type of database.. (this is all one line)

DDLstr = "CREATE TABLE [Students] ([Username] Varchar(20) CONSTRAINT PrimaryKey PRIMARY KEY, [FirstName] Varchar(25), [Surname] Varchar(25))"

Try

mycommand = New OleDb.OleDbCommand(DDLstr, myconnection)

mycommand.ExecuteNonQuery()

MsgBox("Students Table created")

Catch ex As Exception

MsgBox("Students Table not Created " & ex.Message)

End Try

End Sub

Now create the Exams Table, which will hold information about the different exam each subject has

|  |  |
| --- | --- |
| Field | Properties |
| ExamCode | Varchar(6) , Primary Key |
| ExamTitle | Varchar(25) |
| Subject | Varchar(25) |

Now create the Results Table

|  |  |
| --- | --- |
| Field | Properties |
| ResultID | Varchar(6) , Primary Key |
| UserName | Varchar(20) |
| ExamCode | Varchar(6) |
| Score | Int |

We will be adding an extra fields later the 3 tables

Open the access file and look at the tables that have bees created… does everythong look correct?

### Task 5 Adding Fields to a table

Here is an example that adds a field to an existing table:

Sub CreateField()

'Purpose: Illustrates how to add a field to a table using DDL.

Try

DDLstr = "ALTER TABLE [Exams] ADD COLUMN [Weight] REAL"

mycommand = New OleDb.OleDbCommand(DDLstr, myconnection)

mycommand.ExecuteNonQuery()

Msgbox("Field Added Sucessfully")

Catch ex As Exception

Msgbox("Field NOT Added Sucessfully " & ex.Message)

End Try

End Sub

Now add the following:

'DDLstr = "ALTER TABLE [Exams] ADD COLUMN [Weight] REAL"

'DDLstr = "ALTER TABLE [Result] ADD COLUMN [RawScore] REAL"

'Add MaxScore filed to the Exam Table with type integer

'Add a DateofBirth Field to the Student Table

'Add a Gender Field to Student Table

'Add ExamDate to Result (just storing the year as an integer)

### Task 6 Adding data to a table

Let’s start with adding some hard coded data (you will never do this in practice but that way you can see the syntax… (the strings are made smaller to fit on one line… We will look at ways to make you code more readable later)

Sub InsertOneRecord() ‘A literal SQL string

' we are now using SQL rather than DDL

' here is an insertion of one record using some hard coded values

Dim SQLstr As String

Try

SQLstr = "INSERT INTO Students([Username], [FirstName], [Surname]) VALUES ('BB','Robert','theBuilder')"

mycommand = New OleDbCommand(SQLstr, myconnection)

mycommand.ExecuteNonQuery()

Msgbox("inserted")

Catch ex As Exception

Msgbox(ex.Message)

End Try

Make sure you understand the syntax…

* why are bits of the string in square brackets?
* Why are sections in single quotes?

Now we will do the same thing again BUT with data stored in variables

Sub InsertAnotherRecord()

' Building SQL strings with variables takes care..... you need to be alert... the world needs more lerts

Dim SQLstr As String

Dim UName As String = "WB"

Dim fname As String = "Wendy"

Dim sName As String = "theBuilder" ' are bob and wendy married?

Try

'Make sure you REALLY understand how this concatentaion is working!

SQLstr = "INSERT INTO Students([Username], [FirstName], [Surname]) VALUES ('" & UName & "','" & fname & "','" & sName & "')"

mycommand = New OleDbCommand(SQLstr, myconnection)

mycommand.ExecuteNonQuery()

Console.WriteLine("inserted")

Catch ex As Exception

Console.WriteLine(ex.Message)

End Try

End Sub

Alter the Procedure to add the following data to the **EXAMS** Table. The column headings are the exact field names..

| **ExamCode** | **ExamTitle** | **Subject** | **Weight** | **MaxScore** |
| --- | --- | --- | --- | --- |
| COMP999 | Geekdom | Computing | 0.6 | 100 |

Add Textboxes to your MainForm to allow you to enter data into the **Students** and **Exams** Tables.   
(probably best to handle as separate events )

### Task 7 Adding data to the database from CSV files.

It is often useful to have sets of test data stored as text files. As it happens most DBMS also have CSV import… but that would be no fun! It gives us a chance to see a sound way to manipulate files in .net.

Here is an example for the Student database… Make sure this inserts correctly

Sub InserttoStudentFromCsv()

' here is an insertion of a set of records taken from a correctly formatted CSV file.

'You need to have put the CSV files into the Debug bin folder

‘get them here <https://drive.google.com/file/d/0B1nECgSBg0g7S0NDamxOTDFWM0E/view?usp=sharing>

Dim SQLstr As String

Dim SR As New StreamReader("students.csv") 'hope you have this file in the debug bin folder!

Dim txtLine As String

Dim FieldStr As String()

Do Until SR.EndOfStream ' keeps going until end of file

Try

txtLine = SR.ReadLine()

FieldStr = txtLine.Split(",")

SQLstr = "INSERT INTO Students([Username], [FirstName], [Surname]) VALUES ('" & FieldStr(0) & "','" & FieldStr(1) & "','" & FieldStr(2) & "')"

mycommand = New OleDbCommand(SQLstr, myconnection)

mycommand.ExecuteNonQuery()

Console.WriteLine("inserted")

Catch ex As Exception

Console.WriteLine(ex.Message)

End Try

Loop

End Sub

Alter the above code to insert the csv data for the Exams, Users and Results Tables…

' The purpose of this code is to give you a taste of DDL.

' We are using a Microsoft Access MDB file database NOT Ideal but

' very easy for you to see what has happened instantly

' You need to add a referance to the "Microsoft.ADO.Ext 6.0 for Security and DDL" (I don't think the version matters here)

'By Aware MySQL syntax for DDL is slightly different.

Module Module1

Dim mycommand As OleDbCommand

Dim myconnection As OleDbConnection

Dim myReader As OleDbDataReader

Dim DDLstr As String

Sub Main()

CreateDataBase()

OpenDataBase()

CreateStudentTable()

CreateExamTable()

CreateResultTable()

CreateField()

'InsertOneRecord()

Console.ReadLine()

myconnection.Close()

End Sub

Sub OpenDataBase()

Try

myconnection = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Freshtest.mdb ")

myconnection.Open()

Msgbox("Database Opened Successfully")

Catch ex As Exception

Msgbox("Database not opened " & ex.Message)

End Try

End Sub

Sub CreateDataBase()

'This isn't DDL at all really... But is needed if we are using an MDB file

Dim cat As Catalog = New Catalog()

Try

cat.Create("Provider=Microsoft.Jet.OLEDB.4.0;" & \_

"Data Source=DDLTest.mdb;" & \_

"Jet OLEDB:Engine Type=5")

Msgbox("Database Created Successfully")

Catch ex As Exception

Msgbox("Database Creation failed " & ex.Message)

End Try

cat = Nothing

End Sub

Sub CreateStudentTable()

DDLstr = "CREATE TABLE [Students] ([Username] Varchar(20) CONSTRAINT PrimaryKey PRIMARY KEY, [FirstName] Varchar(25), [Surname] Varchar(25))"

Try

mycommand = New OleDb.OleDbCommand(DDLstr, myconnection)

mycommand.ExecuteNonQuery()

Msgbox("Students Table created")

Catch ex As Exception

Msgbox("Students Table not Created " & ex.Message)

End Try

End Sub

Sub CreateExamTable()

'we are deliberatly leaving out the weight of each exam for the moment

DDLstr = "CREATE TABLE [Exams] ([ExamCode] Varchar(6) CONSTRAINT PrimaryKey PRIMARY KEY, [ExamTitle] Varchar(25), [Subject] Varchar(25))"

Try

mycommand = New OleDb.OleDbCommand(DDLstr, myconnection)

mycommand.ExecuteNonQuery()

Msgbox("Exam Table created")

Catch ex As Exception

Msgbox(" Exam Table not Created " & ex.Message)

End Try

End Sub

Sub CreateResultTable()

'we are deliberatly leaving out RawScore for the moment

DDLstr = "CREATE TABLE [Result] ([ResultID] Varchar(6) CONSTRAINT PrimaryKey PRIMARY KEY, [Username] Varchar(20), [ExamCode] Varchar(6))"

Try

mycommand = New OleDb.OleDbCommand(DDLstr, myconnection)

mycommand.ExecuteNonQuery()

Msgbox("Result Table created")

Catch ex As Exception

Msgbox("Result Table not Created " & ex.Message)

End Try

End Sub

Sub CreateNotesTable()

'use the syntax abaove to create a notes table that will be used to store specific informtaion about a studetn sitting an exam

' with the following fields: NotesID INT, StudentUserName VarChar, ExamCode VarChar, NoteTxt Varchar

End Sub

Sub CreateField()

'Purpose: Illustrates how to add a field to a table using DDL.

'DDLstr = "ALTER TABLE [Exams] ADD COLUMN [Weight] REAL"

'DDLstr = "ALTER TABLE [Result] ADD COLUMN [RawScore] REAL"

'Add MaxScore filed to the Exam Table with type integer

'Add a DateofBirth Field to the Student Table

'Add a Gender Field to Student Table

'Add ExamDate to Result (just storing the year as an integer)

Try

DDLstr = "ALTER TABLE [Exams] ADD COLUMN [Weight] REAL"

mycommand = New OleDb.OleDbCommand(DDLstr, myconnection)

mycommand.ExecuteNonQuery()

Msgbox("Field Added Sucessfully")

Catch ex As Exception

Msgbox("Field NOT Added Sucessfully " & ex.Message)

End Try

End Sub

Sub DropField()

'Purpose: Delete a field from a table using DDL.

'Change the DDL string to remove the

Try

DDLstr = "ALTER TABLE [MyTable] DROP COLUMN [DeleteMe];"

mycommand = New OleDb.OleDbCommand(DDLstr, myconnection)

mycommand.ExecuteNonQuery()

Msgbox("Field Deleted Sucessfully")

Catch ex As Exception

Msgbox("Field NOT Deleted Sucessfully " & ex.Message)

End Try

'Use the above commands to ensure you have the following tblaes and fileds exactly like this:

End Sub

Sub InsertOneRecord()

' we are now using SQL rather than DDL

' here is an insertion of one record using some hard coded values

Dim SQLstr As String

Dim fname As String = "wendy"

Try

'SQLstr = "INSERT INTO Students([Username], [FirstName], [Surname]) VALUES ('BB','Robert','theBuilder')"

SQLstr = "INSERT INTO Students([Username], [FirstName], [Surname]) VALUES ('WB','" & fname & "','theBuilder')"

mycommand = New OleDbCommand(SQLstr, myconnection)

mycommand.ExecuteNonQuery()

Msgbox("inserted")

Catch ex As Exception

Msgbox(ex.Message)

End Try

End Sub

Sub InsertFromCsv()

' here is an insertion of a set of records taken from a correctly formatted CSV file.

'You need to have put the CSV files into the Debug bin folder

Dim SQLstr As String

Dim SR As New StreamReader("students.csv")

Dim txtLine As String

Dim Strings As String()

txtLine = SR.ReadLine()

Strings = txtLine.Split(",")

Try

'SQLstr = "INSERT INTO Students([Username], [FirstName], [Surname]) VALUES ('BB','Robert','theBuilder')"

mycommand = New OleDbCommand(SQLstr, myconnection)

mycommand.ExecuteNonQuery()

Msgbox("inserted")

Catch ex As Exception

Msgbox(ex.Message)

End Try

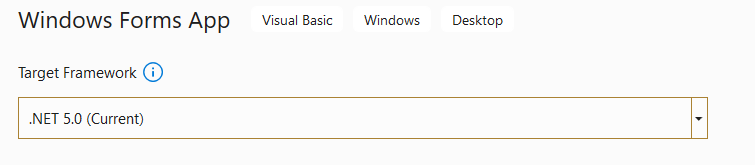
End Sub

End Module

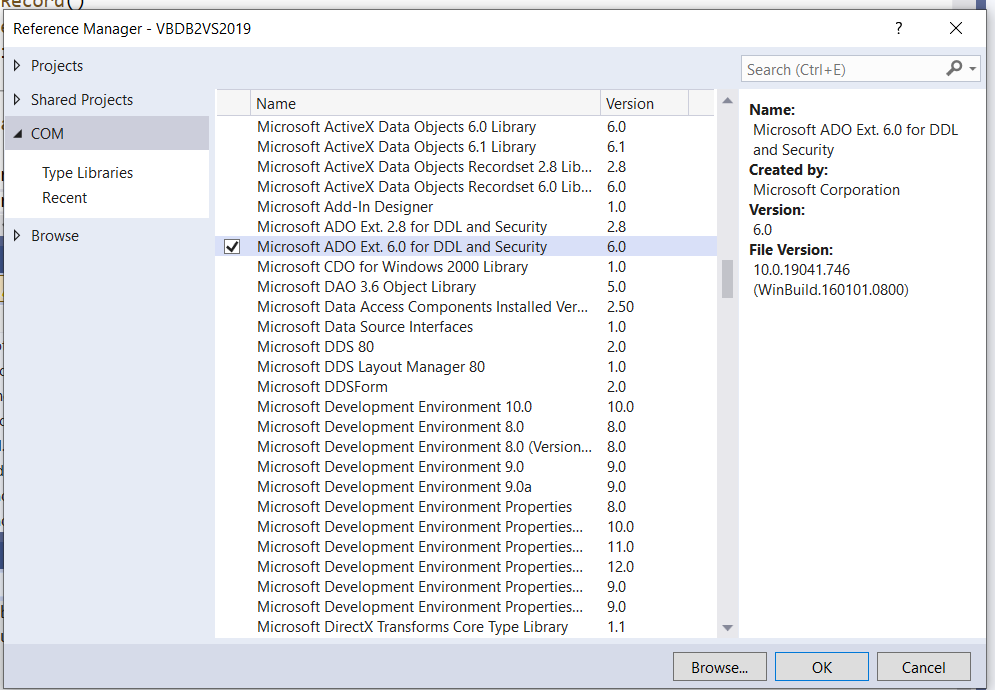
Use .Net 5.0 rather than Core 3.0 or 3.1

## Working with VS 2019

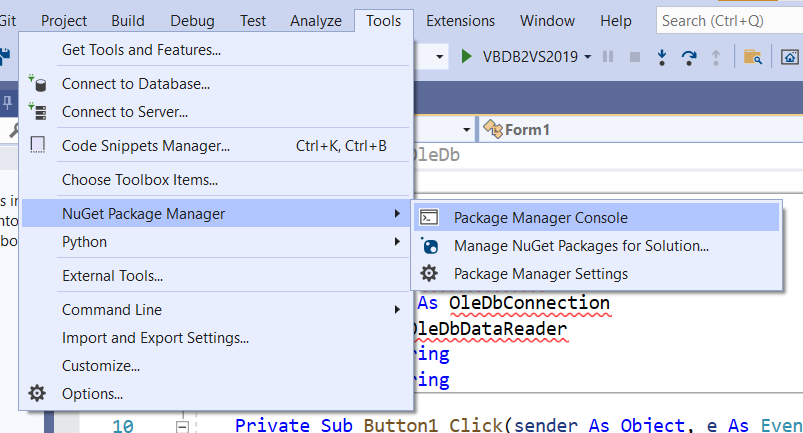
1. Use .NET Framework 5.0



2. For ADOX use v 6.0

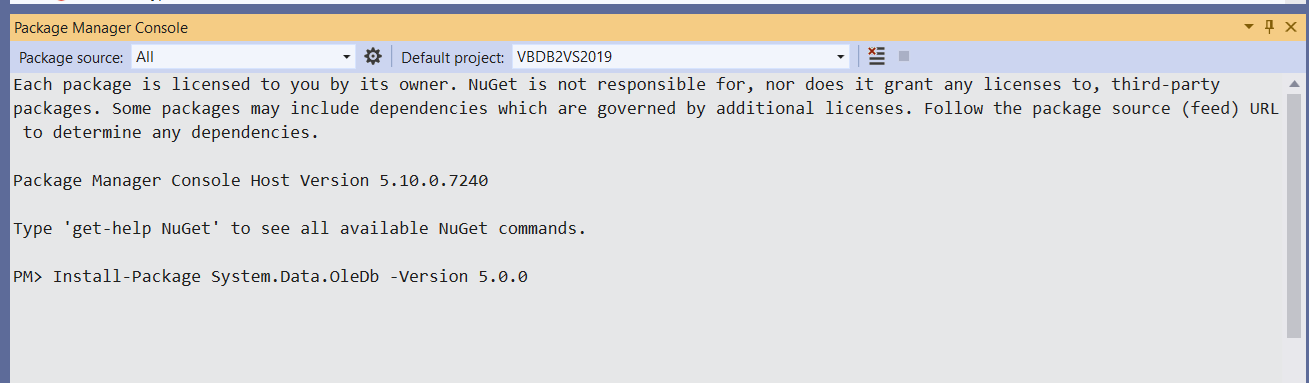


3. For the relevant OleDb library use the the NuGet package manager:



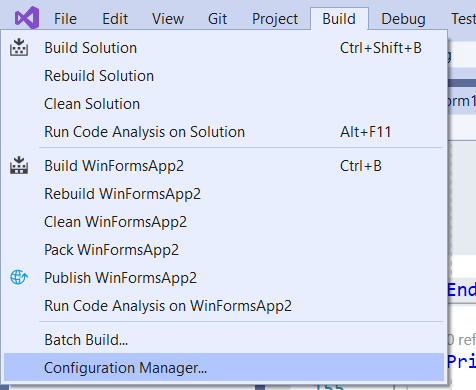
And run the following in the package manager console window:

Install-Package System.Data.OleDb -Version 5.0.0

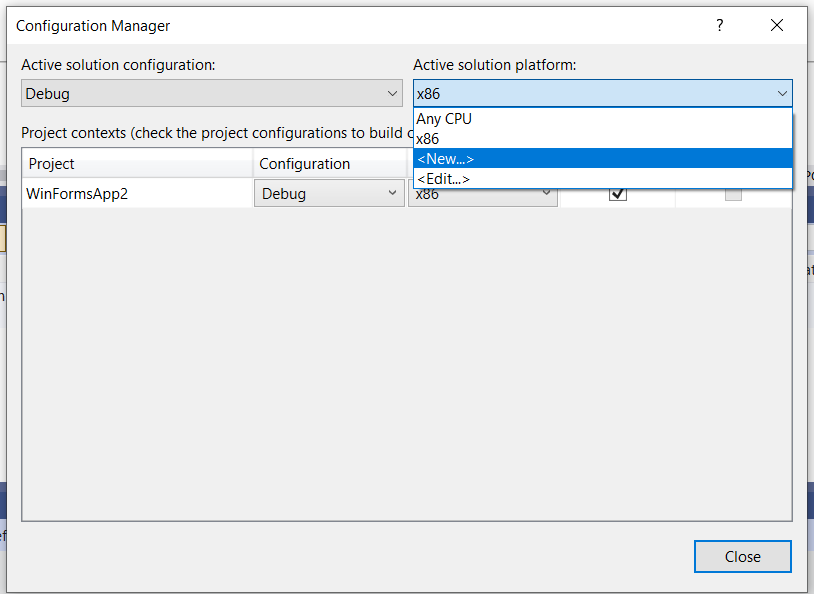


4. Change the visual studio project to target 32-bit rather than 64-bit

Use configuration manager:



Choose the x86 configuration (choose <new> and add x86 if it isn’t in the list)



5. Finally change all atabase connection strings to the following:

“Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Test.mdb”