# *TinyDB*® Users Guide

***What is TinyDB ?***
TinyDB gives database ease of use and power when you don't need a large database engine. Excellent for replacing random files or text files used for storing data. TinyDB is an excellent choice for projects which typically contain less than 20,000 records. TinyDB can store up to *2 billion* records, but the real power is in smaller database applications where size and speed are major considerations. TinyDB adds ***only 39k*** to your application as opposed to megabytes worth of DLLs required for other database engines.

***How do I use TinyDB ?***
TinyDB is extremely easy to use and its events and properties are Similiar to ones that you are already using so the learning curve is minimal. TinyDB installs through the Tools/References menu and when selected adds a new class of TinyDB objects to your development environment.

***How Does TinyDB work, what are the advantages of TinyDB ?***
TinyDB is actually a small DBMS that uses popular properties and methods to add, edit and find data in a structured format. TinyDB lets you retrieve or set fields by the field name making it much easier to use than binary or random files where you'd have to create a non-flexible structure or keep track of record numbers. TinyDB uses a fast byte mode search which can scan thousands of record in seconds. TinyDB contains a series of search parameters not found in many other larger database engines such as Startslike, EndsLike, HAS, Startsless, Endsless and many more. Searching for string data has never been easier. TinyDB adds ***only 39k*** to your application and is extremely fast. TinyDB is an excellent tool where you want database ease but don't want the overhead of a large database engine. Think of all the applications that typically only store 300 or 400 records such as phone dialers, to do lists, small pims etc., why install a database that takes up 5 meg or more for simple apps.

***What about Multi-User environments ?***
**TinyDB is Multi-User capable** and opens files in shared mode so that many users or applications can open the TinyDB database files at the same time.

***How do I purchase TinyDB ?***
‘TinyDB is currently available by two methods:
‘
‘CIS SWREG #10390:  The cost for this method is $69.00(US). You will receive TinyDB for VB5 ‘documentation emailed to you via CIS within 24 hours.
‘
‘US Postal Service by Check or Money Order only: The cost for this method is $30.00(US). You ‘will receive both the TinyDB 16 bit and TinyDB 32 bit DLLs/exes and printed documentation ‘mailed to you the same day we receive your order.
‘
‘For additional information you can contact Worksaver Software by email at CIS: 71662,467 or ‘worksavr@ultranet.com Other availability and ordering options will be mailed to you as soon as ‘they are available.

## Installing TinyDB:

If your tinyDB was downloaded and is in a zip file format then follow these steps otherwise proceed to the from floppy section.

The zip file contains installation and documentation for TinyDB5. Follow these easy steps to install TinyDB.

1) Create a directory where you would like the TinyDB files to be placed ex: C:\TINYDB
2) Unzip the distribution file ex: Pkunzip -d A:\tinydb.zip c:\tinydb  (make sure you use the -d option to create the install sub directories)
This will create three sub directories under the directory you created, these will be Install , Docs and Samples. Install contains the distribution for TINYDB.

From Floppy:
**To install TinyDB for VB 5** run setup.exe that is contained in the Install directory, this is a very short setup procedure and does not ask you for a root directory, it installs TinyDB5.DLL  into your windows/system sub directory and registers it.

## Using TinyDB in the VB Programming Environment:

In the VB IDE you will have to tell your project that it's using the objects made available by TinyDB to do this select Project,  References from the VB main menu. When the references window pops up scroll through the list and find TinyDB for Visual Basic 5 make sure it's checked, then select OK.

## Declaring the TinyDB object in your application

TinyDB being an object with properties and methods has to be declared for VB code to recognize and use it. You can do this in several methods as described below:

**MYDB in the following examples can be substituted for any valid variable name.**

To Declare **Globally** so all procedures have access to the object,
Insert this line into a .BAS files declaration section:
```
GLOBAL MYDB as NEW TINYDB   ' declares the TinyDB engine as object MYDB
```

To Declare so only the **form** that this is declared on has access to TinyDB,
Insert this line in the forms declaration section:
```
PRIVATE MYDB as NEW TINYDB        ' declares the TinyDB engine with no code from
                                  ' others forms being able to access it

PUBLIC MYDB as NEW TINYDB         ' declares the TinyDB engine with code from
                                  ' others forms being able to access it
```

At the **procedure level**, insert one of the following lines into your function or sub:
```
DIM MYDB as NEW TINYDB            ' declares the TinyDB engine available only to this
                                  ' procedure
```

## Connections and Multiple Databases

TinyDB is Multi-User capable but there are no facilities for record locking.

Once a TinyDB Object has been declared in one of the above methods you can begin to use the properties and procedures listed in the next section.

Each database that you wish to have open at the same time needs its own declaration of the TinyDB object for example to open two databases it would be declared in this fashion:

```
Global Database1 as new TinyDB
Global Database2 as new TinyDB

x=Database1.startdb("c:\db\db1.sdb)
```

```
x=Database2.startdb("c:\db\db2.sdb)
```

You can also have more than one connection to the same database:

```
Global Database1 as new TinyDB
Global Database2 as new TinyDB

x=Database1.startdb("c:\db\db1.sdb)
x=Database2.startdb("c:\db\db1.sdb)
```

## File Naming Conventions

Although TINYDB does not have a default file extension. You will find in the examples that the default extension used is .SDB  However you can use whatever extensions or file names that you like with TinyDB as long as they conform to standard file naming cenventions. TinyDB supports long filenames.

TinyDB® Methods and Properties:

## AddField Method

Type: **Subroutine**
MyDB.AddField FieldName, FieldType, FieldLength(Optional)

AddField adds a new field definition when creating a new database. You must execute StartCreate before using AddField. There is a 255 field limit when adding fields to TinyDB.

AddField uses the following parameters:

FieldName: The name of the database field to create, up to 25 characters.
FieldType: One of the following valid field types below

0 = Text ,max length 32767
1 = Integer ,length =2
2 = Long ,length =4
3 = Single, length =4
4 = Double, length =8
5 = Currency, length =8
6 = Date/time, length =8
7 = Byte, length =1

Refer to VB On-line help for more info on data types.

FieldLength: The length of the field, ignored on all types but type 0 (text)

Syntax:
```
MyDB.AddField "NAME", 0, 35     ' Create a field "Name", 35 characters long.
```

Returns: Nothing

See CreateDB for more information on using the AddField method.

## AddNew Method

Type: **Subroutine**
Adds a new record to the Database

Syntax:
```
MyDB.AddNew                ' Creates a blank record and makes it the
                           ' current record.
```

Returns: Nothing

## BOF Property

BOF

The BOF Property is set to true when any of the move or find methods place the current record at the beginning of the database file or record 0.

Syntax:
```
If not Mydb.BOF then MyDB.MovePrevious     ' Check for bof before
                                           ' attempting
                                           ' MovePrevious
```

Returns: Integer

## CreateDB Method
Type: **Function**
Result = MyDB.CreateDB(DatabaseName)
Creates a new database that has been defined by AddField items. You must call the StartCreate method before using CreateDB.

Example:
```
Dim MyDB as new TinyDB
MyDB.StartCreate
MyDB.AddField "Name", 0, 35
MyDB.AddField "Company", 0, 35
MyDB.AddField "Address1", 0, 35
MyDB.AddField "Address2", 0, 35
MyDB.AddField "City", 0, 35
MyDB.AddField "State", 0, 2
MyDB.AddField "Zip", 0, 15
result = MyDB.CreateDb( App.path+"\Sample.SDB" )
If result = False Then MsgBox "Could not create file."
```

**Returns:** Integer

## DatabaseName Property
Returns the name of the currently open database.

Syntax: X$ = MyDB.DatabaseName to retrieve the database name.

See CreateDB for more information.

**Returns:** String

## DBOpen Property
DBOpen

The DBOpen property is set to true if the last attempt at Starting a dtabase was successful, false if no database is open.

Syntax: If MyDb.DBOpen = False then MsgBox "No Database is open."

Returns: Integer

## Delete Method
Type: **Subroutine**
MyDB.Delete Deletes the current record.
The Delete method does not physically delete the record but marks the records deleted flag to 1 (0= not deleted).

Syntax: MyDB.Delete

Returns: Nothing

## Deleted Property

Deleted (See notes section)
Show the status of the deleted flag of the current record. You can also use the deleted flag to find deleted record for adding new records instead of executing AddNew. such as:

```
MyDB.FindFirst "DeletedFlag", "=", 1      ' Finds the first deleted record and makes it the current
                                    ' record.
```

```
Syntax:                   If MyDB.Deleted then MyDB.MoveNext
```

Returns: Integer

## EOF Property

EOF

The EOF Property is set to true when any of the move or find methods place the current record at the last record of the database.

```
Syntax:                   If not MyDB.EOF then MyDB.MoveNext          ' Check for EOF before
                                                                      ' attempting MoveNext
```

## Field Method

Type: **Function**
*Variant* = Field(FieldName)
Sets or returns a value from the specified field from the current database record.

The following segment is an example of **retrieving** field data.

```
Dim MyDB as new TinyDB
MyDB.Startdb App.Path + "\Sample.sdb"
If MyDb.DBOpen = False Then MsgBox "Could not open Database."
MyDB.MoveFirst
Text2(0) = MyDB.Field("Name")
Text2(1) = MyDB.Field("Address1")
Text2(2) = MyDB.Field("Address2")
Text2(3) = MyDB.Field("City")
Text2(4) = MyDB.Field("State")
Text2(5) = MyDB.Field("Zip")
```

The following segment is an example of **Saving** field data.

```
MyDB.Field("Name") = Text2(0)
MyDB.Field("Address1") = Text2(1)
MyDB.Field("Address2") = Text2(2)
MyDB.Field("City") = Text2(3)
MyDB.Field("State") = Text2(4)
MyDB.Field("Zip") = Text2(5)
```

**Returns:** Variant (Defined by field type)

## FieldCount Property

FieldCount

The FieldCount property returns the number of fields currently defined in the database.
The FieldCount is 1 based.

Syntax: X = MyDB.FieldCount

Returns: Integer

## FieldName Property

FieldName(FieldNumber as Integer)

The FieldName property returns the name of the field specified by the fieldnumber index.
The field count is 1 based so the first field is field 1, not field 0.

Syntax: X$ = MyDB.FieldName(1) ' Sets X$ to the first field name

Example of how to show all fields in unknown database:

```
For a = 0 To MyDB.fieldcount
text1.Text = text1 + "Name: " + MyDB.Fieldname(a) + vbCrLf
text1.Text = text1 + "Type: " + Str$(MyDB.Fieldtype(a)) + vbCrLf
text1.Text = text1 + "Length: " + Str$(MyDB.Fieldlength(a)) + vbCrLf + vbCrLf
Next a
```

**Returns:** String

## FieldLength Property

FieldLength(FieldNumber as Integer)

The FieldLength Property returns the length of the field specified by the fieldnumber index.

Syntax: X = MyDB.Fieldlength(0) ' Sets X = to the length of Field 0

Returns: Integer

## FieldType Property

FieldType(FieldNumber as Integer)

The FieldType Property returns the type of the field specified in the Fieldnumber index.

Syntax: X = MyDB.FieldType(0) ' Sets X to the corresponding field type
' (see AddField for details on Field Types)

**Returns:** Integer

## FindFirst Method

Type: **Function**
x = FindFirst (FieldName, Comparison, SearchItem)

FindFirst finds the first occurance of SearchItem in the specified field of the currently open Database. The FindFirst method is case insensitive. If no match is found the **NoMatch** property is set to True.

' would find the first occurance of a record where Name started like 'Mark'

Example: if MyDB.FindFirst( "Name", "StartsLike", "Mark") then Msgbox "No Matching Records."

The following with work with any of the data types:

< less than
> greater than
<= less or equal
>= Greater or equal
<> not equal
= equals

The following will work with String fields only:

| | |
|---|---|
| HasLike | ' Search for occurance of SearchItem |
| StartsLike | ' Compares only the starting length of the field which is equivalant to the ' length of SearchItem. |
| EndsLike | ' Compares only the end x characters of the field which is equivalant to ' the length of SearchItem. |
| EndGreater | ' Similiar to EndsLike except looks for a greater than result |
| StartGreater | ' Similiar to StartsLike except looks for a greater than result |
| StartLess | ' Similiar to StartsLike except looks for a less than result |
| EndLess | ' Similiar to EndsLike except looks for a lass than result |

When using any of these operators in string searches, the SearchItem and the item being compared to are both converted to uppercase and trimmed of beginning and trailing spaces before being compared.

Returns: Integer, true if data record was found, false if not matching records found.

## FindNext Method

Type: **Function**
x = MyDB.FindNext

FindNext will try and locate the next field that matches the search parameters setup by a previous call to FindFirst. If no match is found the **NoMatch** property is set to True.

Syntax: if Not MyDB.FindNext then MsgBox "No Matching Records."

Returns: Integer, true if data record was found, false if not matching records found.

## FindPrevious Method

Type: **Function**
x = MyDB.FindNext

FindNext will try and locate the previous field that matches the search parameters setup by a previous call to FindFirst. If no match if found the **NoMatch** property is set to True.

Syntax: if Not MyDB.FindPrevious then MsgBox "No Matching Records."

Returns: Integer, true if data record was found, false if not matching records found.

## MoveFirst Method
Type: **Subroutine**
MoveFirst

MoveFirst movesto the first record of the currently opened database. MoveFirst also sets the **BOF** flag to true.

Syntax:        MyDB.MoveFirst

## MoveLast Method
Type: **Subroutine**
MoveLast

MoveLast moves to the last record of the currently opened database. MoveLast also sets the **EOF** flag to true.

Syntax:        MyDB.MoveLast

## MoveNext Method
Type: **Subroutine**
MoveNext

MoveNext moves to the Next record of the currently opened database. MoveNext will also set the **BOF** and **EOF** properties if they are true.

Syntax:        If Not MyDB.EOF then MyDB.MoveNext

## MovePrevious Method
Type: **Subroutine**
MovePrevious

MovePrevious moves to the previous record of the currently opened database. MovePrevious will  also set the **BOF** and **EOF** properties if they are true.

Syntax:        If Not MyDB.BOF then MyDB.MovePrevious

## NoMatch Property
Nomatch

The Nomatch property is set after on of the Find methods is executed and set to true if no match was found, false if a match was found.

Example:        If MyDB.NoMatch then MsgBox "No Matching Records."

## Record Property
Record

Sets or returns the record number of the current record.

```
Syntax:     X = MyDB.Record      ' Set X to the current record number.
            MyDB.Record = X      ' Sets the current record number to X
```

**Returns:** Long Integer

## RecordCount Property

RecordCount

Returns the current number of records the database contains. Records are 0 based, the first record is 0, the second is 1 and so on. The last record in the database will have an actual record number of RecordCount-1.

```
Syntax:                 X=MyDB.RecordCount          ' Sets X to the number of records in the
                                                    ' currently opened database.
```

**Returns:** Long Integer

## StartCreate Method

Type: **Subroutine**
StartCreate

Starts the creation process of creating a NEW database.

```
Syntax:                 MyDB.StartCreate            ' Starts Creating a new database
```

The following Example creates a database Sample.sdb:

```
Dim MyDB as new TinyDB
MyDB.StartCreate
MyDB.AddField "Name", 0, 35
MyDB.AddField "Company", 0, 35
MyDB.AddField "Address1", 0, 35
MyDB.AddField "Address2", 0, 35
MyDB.AddField "City", 0, 35
MyDB.AddField "State", 0, 2
MyDB.AddField "Zip", 0, 15
Result = MyDB.CreateDb(App.path+"\Sample.SDB")
If result = False Then MsgBox "Could not create file."
```

See AddField and CreateDB for more information on creating new databases.

## StartDB Method

Type: **Function**
x = MyDB.StartDB DatabasePath

The StartDB methods opens the database and puts the record pointer on record 0. StartDB also sets the **DBOpen** flag to true if the open was successful.

Example:

```
If MyDb.DBOpen then MsgBox "Database is already open."  ' db is already open
If Not MyDB.Startdb( App.Path + "\Sample.sdb") Then MsgBox "Could not open Database."
```

Returns: Integer

## StopDB Method

Type: **Subroutine**

The StopDB method closes the current database and sets the **DBOpen** flag to False.

Syntax:  MyDB.StopDb

Returns: Nothing

## Notes:

**Database Considerations:**
Each record in the database contains a field **DELETEDFLAG**, this field is used to mark whether the record is deleted or not. DELETEDFLAG is a byte length field and is set to 1 if deleted, 0 if not deleted.

When defining databases be certain to include all the fields you will need and allow also for future expansion if needed. Once a database is created its structure cannot be changed.

Text Field Length, Text fields are stored as fixed length strings in the database so be certain not to over estimate the space needed to keep db size to a minimum.

## Technical Support

Technical Support can be obtained by CIS Email: 71662,467 or by Internet Email: wssw@splusnet.com