DATABASE

Using **Filters** to display certain information:

**Important** – Using filters only change the way the data is displayed; it **DOES NOT** change the database.

You will work with two(2) properties

* The **Filter property**, which is a **string** property
  + this specifies the text of the current filter for the dataset
  + When filtering is applied to a dataset, only those records that meet a filter’s conditions are available
* The **Filtered property**, which is a boolean property
  + Is used to determine whether a filtering is in effect
  + If set to true, filtering is active and if false, filtering is inactive

NOTE: The field name and the content of the field is **NOT case-sensitive**.

NOTE: The filter property does not allow you to combine **OR** clauses within parentheses with an AND clause. Can’t write it as (Grade = 8 **OR** Grade = 11) AND (Gender = ‘F’).

**Using code to set up a filter:**

**First thing:** Switch the **filtered property** to **false** in order that no pre-existed filter is activated!

**STEP 1**: You need to indicate the **name of the data module** where the ADOTable component is stored. e.g. dmMusicData.tblCD

**STEP 2**: Add the **filter property** to the previous step. e.g. dmMusicData.tblCD.Filter

**STEP 3**: Adding the **condition** you want. e.g. dmMusicData.tblCD.Filter := ‘Artist = ‘‘Creed’’’;

**Remember** to set the **filtered** property to **TRUE**.

e.g dmMusicData.tblCD.Filtered := True;

NOTE: The **filter property is a string** – therefore quotes need to be added to the beginning and the end of the condition. Because the field Artist should contain a string, the value ‘Creed’ should be in quotes. The result is that a string (Creed) should be placed inside another string. In Delphi you have to use two(2) singles quotes for the ‘inside’ string.

e.g. dmMusicData.tblCD.Filter := ‘Amount of CDs = ‘ + edtNumCD.Text; //Integer

OR e.g. dmMusicData.tblCD.Filter := ‘Artist = ‘’’ + edtArtist.Text + “”; //String

OR e.g. dmMusicData.tblCD.Filter := ‘Price > ‘ + edtPrice.Text; //Integer

OR e.g. dmMusicData.tblCD.Filter := ‘CD Name = ‘ + QuotedStr(edtCDName.Text); //String

**Remember** after a filter has been set only the filtered data will be displayed. **To view** the **original set of records**, the **Filtered property** must be set to **False.**

**RecordCount** is a property of a dataset that contains the number of records in the dataset. e.g. lblOutput.Caption := IntToStr(dmMusicData.tblCD.RecordCount);

**CREATE SUMMARIES**

**Perform calculations** based on the contents of the fields of the various records in a table.

**Similar to a text file**, where data can be **read line by line** using code in a **while loop** to iterate through the records one by one.

**A record** in a table has to be the **‘active’ or ‘current’ record**, the invisible pointer is positioned at before changes can be made.

The value in a field in the active record can be obtained using the name of the field.

e.g.

**var** rTotal, rAvg : Real;

**Begin**

rTotal := 0;

dmMusicData.tblCD.Open; //Sets the Active property for this table to True, to read from it

dmMusicData.tblCD.First; //Pointer points to the first record

**While** NOT dmMusicData.tblCD.Eof do // Execute loop until end of table is reached

**Begin**

rTotal := rTotal + dmMusicData.tblCD[‘Price’]; //Use the value in the field Price

dmMusicData.tblCD.Next; //Move to the next record

**end;**

rAvg := rTotal / dmMusicData.tblCD.RecordCount;

redOutput.clear;

redOutput.Lines.Add(‘Total amount of money: ‘ + FormatFloat(‘R0.00’,rTotal);

redOutput.Lines.Add(‘Average per Learner: ‘ + FormatFloat(‘R0.00’,rAvg);

**end;**

**Remember**, when a filter is applied, only those records will be processed.

SORT RECORDS

Use the sort property to sort the dataset in a specific order.

* Either ASC for ascending OR
* DESC for descending

e.g. tblMusic.Sort := ‘Artist ASC’;

Note – You can sort by two fields, first the one and then the next e.g. tblStudents.Sort := ‘Lname ASC, Fname ASC

FINDING A SPECIFIC RECORD

In order to use the Locate function, you need to **add “DB” to your uses** clause.

Use the Locate method to find specific data. This method **searches the dataset for a specific record** based on a specified value for a particular field and **makes that record the current record.**

Function Locate(KeyFields : String; KeyValues : Variant; Options : TLocateOptions) : Boolean;

//Field //Value //Options

It is a **Boolean function** that returns a value of True or False depending on whether the record is located or not. When found, it returns a True value AND it moves to that field!

**KeyFields** – The field you want to research

**KeyValues** – The value the field should have. If the field is a string, the argument should either be in quotes, or it should be a string variable

**Options** – A set of options. The set is indicated by square brackets [ ]. Separate the different options with a comma.

loCaseInsensitive: Fields and values are matched regardless of the case

loPartialKey: Will search for partial matches starting with the same value.

Example: with dmMusicData do

If tblMusic.Locate(‘Artist’,’Creed’,[loCaseInsensitive]) then

ShowMessage(‘Creed was found’)

Else

ShowMessage(‘Artist not found’);

**When going through all the records in a table**

Remember if the table is not active, you have to code it then: **tblCD.Open**;

STEP 1: Go to the first record: **tblCD.First**;

STEP 2: Loop till end of table: **while NOT tblCD.EOF do**

STEP 3: In the loop, move to next record: **tblCD.Next**;

If for any reason you would like to change the Active property of your table to False, you can simply call the method: **tblCD.Close**; and your table will be Inactive, you won’t be able to read from or write to the table.

DATA MAINTENANCE

A dataset of records – such as a table – is always in a specific state. The state can be changed in the following way:

* Change the State property of the dataset.
* Call a method that automatically places the dataset into a particular state.
* Change certain properties of the ADOTable component.

The default state is set to Browse when you have made your connection.

The two(2) states we will look at are Edit and Insert.

**Insert** – allows a new record to be added to the database

Example:

with dmMusicData do

Begin

tblMusic.Last; //put the pointer at the last record of your dataset

iNo := tblMusic[‘CD\_Number’]; //assigns the value of the primary key to a variable

tblMusic.Insert; //change the state of the dataset to Insert

tblMusic[‘CD\_Number’] := iNo + 1; //adding 1 to the previous number in the dataset

tblMusic[‘CD\_Name’] := edtName.Text;

tblMusic[‘Artist’] := sArtist; //you can make use of an Inputbox to get this string

tblMusic[‘Owner’] := cbbOwners.Items[cbbOwners.ItemIndex];

tblMusic[‘Grade’] := StrToInt(rgpGrade.Items[rgpGrade.ItemIndex]);

tblMusic.Post; //writes the new record to the table

end;

Make sure always to add data validation

**Edit** – allows the content of one or more fields to be modified.

In the following example, every record in one field will be changed: (You can also just a specific record by using the search function (locate).

with dmMusic do

begin

tblMusic.Open;

tblMusic.First;

while NOT tblMusic.EOF do

begin

tblMusic.Edit;

tblMusic[‘Grade’] := tblMusic[‘Grade’] + 1;

tblMusic.Post;

tblMusic.Next;

end;

**Delete** – allows the deletion of one record

You can simply say dmMusic.tblMusic.Delete and it will delete the active (current) record.

However, it is good programming policy to display a user-friendly message to let the user confirm the deletion of the record.

example:

with dmMusic do

begin

sNo := IntToStr(tblMusic[‘CD Number’]);

sName := tblMusic[‘CD Name’];

sArtist := tblMusic[‘Artist’];

If MessageDlg(‘Are you sure you want to delete CD number ‘ + sNo + ‘ of the artist ‘ + sArtist

+ ‘ with the name ‘ + sName + ‘?’, mtWarning, [mbOk,mbCancel],0) = mrOk then

tblMusic.Delete;

end;

* An error will occur if the dataset is in the dsInactive state and the Delete method is executed.
* After the record is deleted the pointer will move to the next record as the active record or if it was the last record that was deleted, the pointer will move to the previous record in the dataset.

Multi Tables

CD.First;

While NOT CD.EOF do

begin

If CD['CDName'] = 'Americana' then

begin

Owner.First;

While NOT Owner.EOF do

begin

if CD['OwnerID'] = Owner['OwnerID'] then

redOutput.Lines.Add(Owner['OwnerFirstName'] + ' is the owner of + CD['CDName'] + ' cd');

Owner.Next;

end;//while Owner

end;//if cdName

CD.Next;

end;//While CD