# Pedigree Cloud

V4

## Form Design

# Table of Contents

# 1 Forms Basics

## 1.1 Entities

A form file contains many templates or forms. Each of these forms corresponds to one or more sheets of paper and a form is a graphical description of how you want to present database data.So basically all your data is in a database file (these in in .DB) and the form files (which end in .FOX) provide reports for displaying that data.

Forms contain drawing entities which can be classified as either static or dynamic. Static entities never change regardless of the data, while dynamic entities are dependent on the data from the database.

**The static entities are:**

| | |
|---|---|
| Text | This does not change when a different entry is selected (unlike fields). Static text is shown in a black colour only. |
| Image | This is a fixed picture (i.e. not linked to the database). Only the file's location is stored in this entity (not the actual picture). Whenever the entity is drawn the file will be read in from the file location and drawn. Images are also used for logos and watermarks. |
| Lines | Lines are coloured in a user selectable colour and width. |
| Rectangles | As for lines. |

**The dynamic entities are:**

| | |
|---|---|
| Records | These appear as green rectangles in the form edit view but will appear as black rectangles in the form view (or not appear at all if their line width is set to invisible). |
| Fields | These appear as blue rectangles in the form edit view. Fields should always be contained within records. |
| Pictures | These are just one type of field but with a special picture capability. They are similar to images, but instead of the picture file name being stored in the picture it is stored in a field of the database. |
| Links | These appear as red lines with a black blob at one end in the form edit view. Links are used to join records (green rectangles). A link should always start in one record and end in another. |

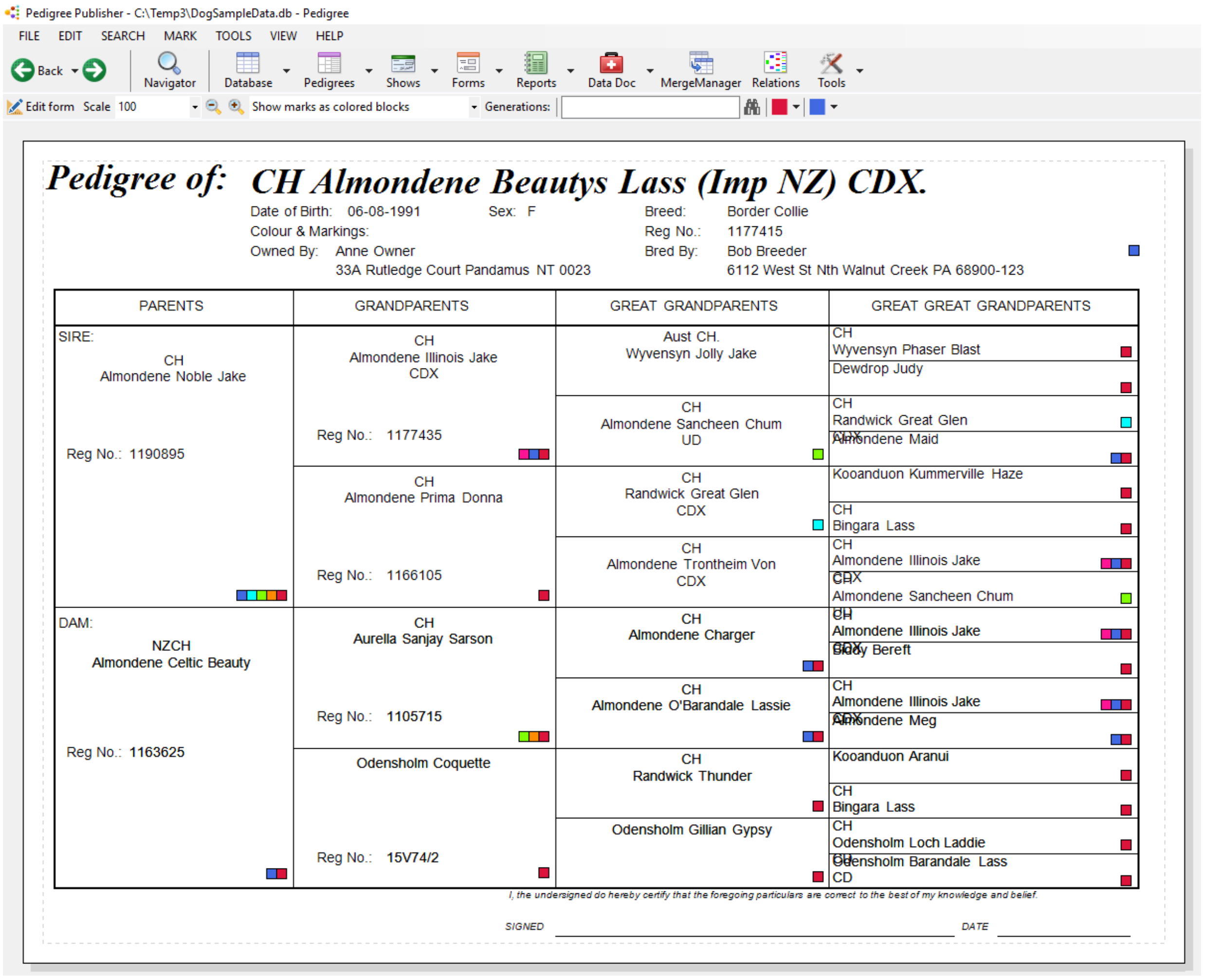
The size of the form, i.e. the size of the paper is set via the **FILE-Print Setup** command.

To edit a form first go to the form view and select the form you wish to edit from the Pedigrees menu.

Next select the edit mode by clicking on the Edit form button.

An example of a form in form view is shown in Figure 2.1.

## Figure 2.1


Pedigree Publisher - C:\Temp3\DogSampleData.db - Pedigree
FILE EDIT SEARCH MARK TOOLS VIEW HELP
Back
Navigator
Database
Pedigrees
Shows
Forms
Reports
Data Doc
MergeManager
Relations
Tools
Edit form
Scale 100
Show marks as colored blocks
Generations:

**Pedigree of: CH Almondene Beautys Lass (Imp NZ) CDX.**

Date of Birth: 06-08-1991 Sex: F Breed: Border Collie
Colour & Markings: Reg No.: 1177415
Owned By: Anne Owner Bred By: Bob Breeder
33A Rutledge Court Pandamus NT 0023 6112 West St Nth Walnut Creek PA 68900-123

| PARENTS | GRANDPARENTS | GREAT GRANDPARENTS | GREAT GREAT GRANDPARENTS |
|---|---|---|---|
| SIRE: CH Almondene Noble Jake Reg No.: 1190895 | CH Almondene Illinois Jake CDX Reg No.: 1177435 | Aust CH. Wyvensyn Jolly Jake | CH Wyvensyn Phaser Blast |
| | | | Dewdrop Judy |
| | | CH Almondene Sancheen Chum UD | CH Randwick Great Glen CDX |
| | | | Almondene Maid |
| | CH Almondene Prima Donna Reg No.: 1166105 | CH Randwick Great Glen CDX | Kooanduon Kummerville Haze |
| | | | CH Bingara Lass |
| | | CH Almondene Trontheim Von CDX | CH Almondene Illinois Jake CDX |
| | | | CH Almondene Sancheen Chum UD |
| DAM: NZCH Almondene Celtic Beauty Reg No.: 1163625 | CH Aurella Sanjay Sarson Reg No.: 1105715 | CH Almondene Charger | CH Almondene Illinois Jake CDX |
| | | | Biddy Bereft |
| | | CH Almondene O'Barandale Lassie | CH Almondene Illinois Jake CDX |
| | | | Almondene Meg |
| | Odensholm Coquette Reg No.: 15V74/2 | CH Randwick Thunder | Kooanduon Aranui |
| | | | CH Bingara Lass |
| | | Odensholm Gillian Gypsy | CH Odensholm Loch Laddie |
| | | | CH Odensholm Barandale Lass CD |

*I, the undersigned do hereby certify that the foregoing particulars are correct to the best of my knowledge and belief.*

*SIGNED* ______________________ *DATE* ____________

The form view shown above is of a four generation pedigree certificate. Items such as "Date of Birth" and "GRANDPARENTS" are examples of text. These are of course static and do not change when different data is selected to be displayed.

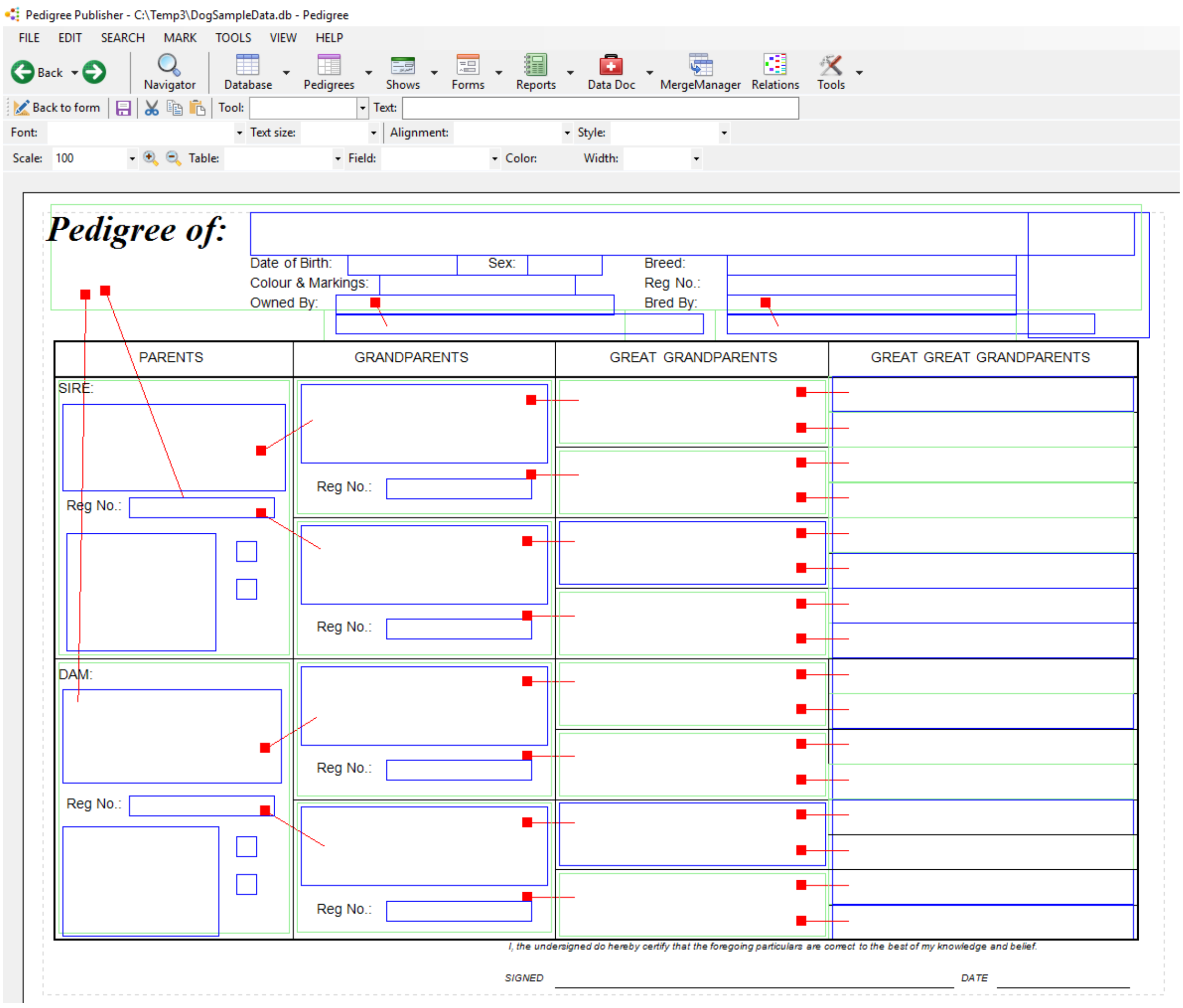
The large rectangle enclosing the four generations is an example of a rectangle. Again it is static.

The lines which divide up the pedigree are an example of lines, which again are static.

The name of the entry "Almondene Beautys Lass" is an example of a field. It will change with the data. Other fields can also be seen, e.g. DOB, "Colour & Markings", "Reg No." etc.

The records and links cannot be seen in the forms view but if we switch to design view as shown in Figure 2.2, then they would appear as green rectangles and red lines respectively.

**Figure 2.2**



*Pedigree of:*

Date of Birth: Sex: Breed:
Colour & Markings: Reg No.:
Owned By: Bred By:

| PARENTS | GRANDPARENTS | GREAT GRANDPARENTS | GREAT GREAT GRANDPARENTS |
|---|---|---|---|
| SIRE: Reg No.: | Reg No.: | | |
| | Reg No.: | | |
| DAM: Reg No.: | Reg No.: | | |
| | Reg No.: | | |

*I, the undersigned do hereby certify that the foregoing particulars are correct to the best of my knowledge and belief.*

*SIGNED* ____________ *DATE* ____________

## 1.2 Text Properties

Text and field entities as described above both have text properties as described in the following subsections.

| | |
|---|---|
| Font | The font is the character font and it may be any of the TrueType fonts installed on your machine. Note that if you design a form using a TrueType font that exists on your machine and then supply your form to someone else who doesn't have that font, then a substitute font will be supplied - it may not appear exactly as intended. |
| Text size | This is the font size in units of 0.1mm. A normal size to use in most forms is 28. Large text could be 36, or 48. The user is free to set any integer font size up to the Windows maximum. |
| Style | This is either, regular, bold, italic or underline. |
| Color | This is the text color. |

## 1.3 Line Properties

Line properties apply to all links, records, fields, lines and rectangles.

| | |
|---|---|
| Width | The line width. |
| Color | The line color. |

## 1.4 Display Order

When drawing, PedPub will draw entities in the order in which they appear in the display list. Every entity is stored in the display list and when a new entity is added to the drawing it is added to the end of the display list, i.e. it is drawn last. An item drawn last will appear above an item drawn before it.

The display order can be changed by selecting items and using EDIT-Arrange-Send to back etc.

## 1.5 Dynamic Entities

The dynamic entities, i.e. records, fields and links are simply a graphical means of performing database queries.

Here are some simple rules which will assist in understanding these concepts:

- A record appears as a green rectangle in edit mode. It corresponds to one record in the database in one particular table.
- The table which each record belongs to is set and can be changed, via the edit box.
- Every form has one and only one root record. It is distinguished by being the only record which is not the target of a link. The actual record that a root record points to is always the current record in that table. The current record is simply the record that is currently highlighted in the database view.
- Every record except the root record is back to the root record either directly or through intermediate records which are themselves linked to the root record.
- Records are linked together by links. A link is directional, it has a source and a destination. The source has a square solid blob on it, the destination end does not.
- Each link has one and only one field set for it. The action of a link is to look up the nominated field from the source record, in the primary field of the destination record, e.g. a link can be from the Owner field of a Pedigree record to a record in the Contacts table.
- Fields appear as blue rectangles in edit mode. A record can contain any number of fields. The fields are simply set to a particular field within the record in which they are contained.

## 1.6 Creating Forms Files

A forms file is a collection of forms. You may wish to create your own forms file to collect together all the forms that you normally use and also to add your own forms.

To create a forms file use **FILE-Create Blank Form File**.

# 2 Forms Tutorial

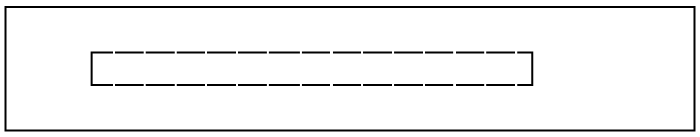
## 2.1 Creating a form file

Normally you would just add a form to an existing form file so you can quickly change between your new form and existing forms, but if you did want to do it use **FILE-Create Blank Form File**.

## 2.2 Designing the Simplest Form

The simplest possible form in design view is shown in Figure 3.1 where the outer rectangle is a record (green color) and the inner rectangle is a field (blue color).

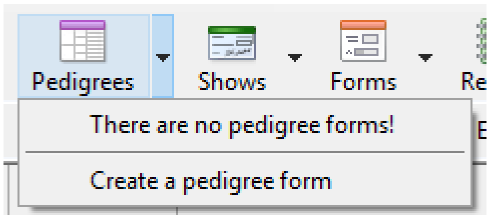
**Figure 3.1**



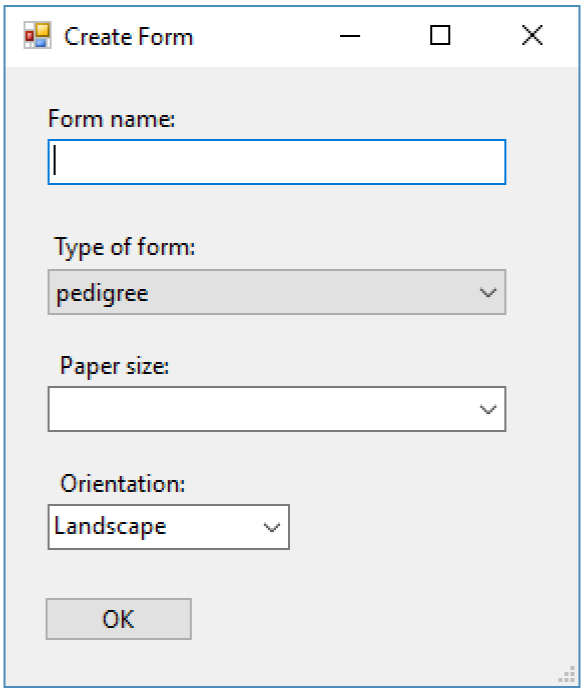
If the record's Table property is set to Pedigree, and the field's Field property is Name then when the form is run it will display the Name of the current record, i.e. the record you double clicked in the grid view.

To create this form follow these steps:

First create a new blank form by clicking on the drop down button next to the Pedigree button on the main tool bar and select **"Create a pedigree form"** as shown below.



The following dialog appears. Set the form name, page size and orientation. Note because you clicked **"Create a pedigree form"** from the Pedigree form the type of form has been preset to "pedigree" but you can also change for type to the other available forms.



Create Form

Form name:

Type of form: pedigree

Paper size:

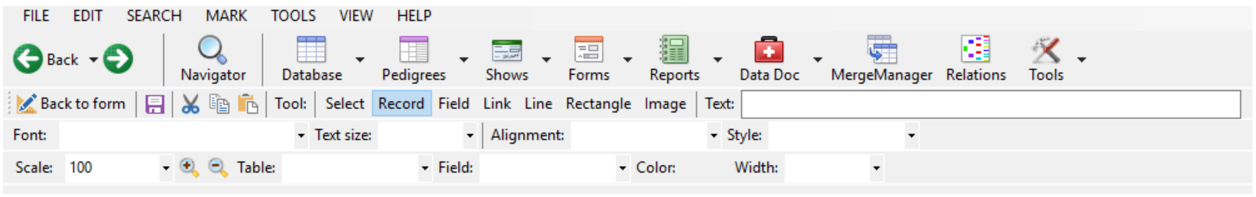
Orientation: Landscape

OK

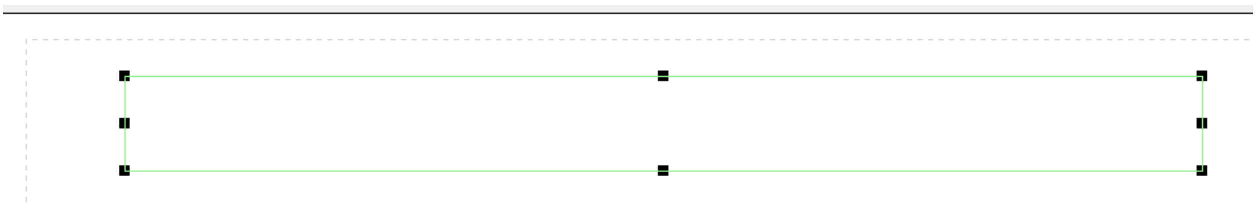
After clicking OK a blank form is shown. Change to the form design view by clicking this button: Edit form

Then select the Record tool button as shown below. Note that when a tool is selected it remains so only until it used once. Once used the current tool reverts to the Select tool. If however the tool button is double clicked, the tool will remain the current tool regardless of how many times it used, until another tool is selected.

With the Record tool selected, click with the mouse, drag and release to describe a rectangle large enough to contain an entry's name. A green rectangle will appear - this is a record.



The form should appear as follows:



Note the black "handles" at the corners and mid point of each line. To resize the record, click in one of those handles, drag and release.

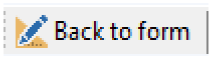
The only property we need to set on the record is which table it belongs to. This is done using the toolbar Table combo. For this exercise select the "Pedigree" table. That completes creating the record.

Now do the same by selecting the Field tool and draw its rectangle within the record. The field's rectangle can be partially outside the record's rectangle but the top left corner must be inside.

The field now has to have what record field it will display. For this exercise select "Name" from the Field combobox. The Font and Text size will also need to be set. In this case set Arial and 20pt.

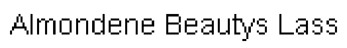
This simple form is now complete. It doesn't do anything more than the display the name of the current subject.

## 2.3 Displaying the Simplest Form

Having completed the forms design, we switch to forms mode by clicking on 

then the form we created would appear as below:

**Figure 3.2**


Almondene Beautys Lass

If we go back to the grid view and double click on another record, its name will be displayed.

The record we added to the form was a **root record** because it was the only record and there were no links into it. Note <u>every form must have one and only one root record</u> - every other record hangs off that record either directly or indirectly.

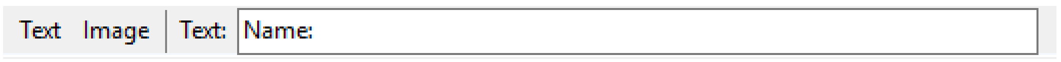
Because the record was set to the Pedigree table, the form will display the current record in the Pedigree table, which happened to be "Almondene Beauty's Lass".

The field was set to a value of "Name", hence in forms view it was replaced with the contents of the name field from the current record.

## 2.4 Adding Static Text

So far we are only displaying dynamic text from the database. We will now add some static text in front of the dymanic data to act as a label. Since the field we are displaying is the name field, the text we will insert is "Name:".

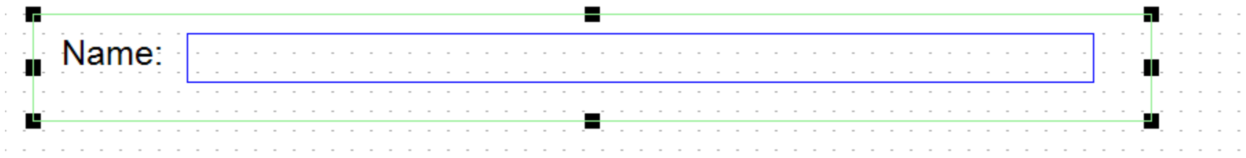
To add the text click on the Text tool button on the toolbar, then click in the form at the top left of the area you want the text to appear, then drag the mouse and release at the bottom right of that area. Next click in the Text: textbox and type the text – see screen shot below.



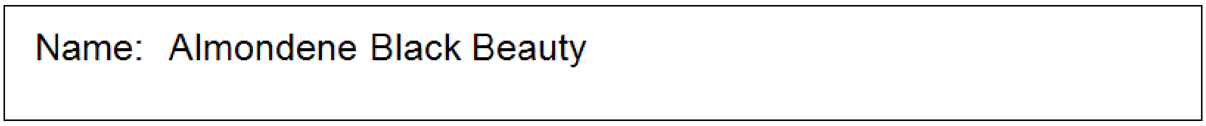
Text: Name:

As the text is typed it will appear in the area previously described. Note the text in the form is still selected because it should be showing its solid black square "handles" at the corners and midpoints. You can therefore move the text by clicking in it and dragging then releasing. It can also be resized by clicking in a handle then dragging and releasing.

The form up to this point should look like this in design view:



And like this in form view:


Name: Almondene Black Beauty

## 2.5 Adding More Fields

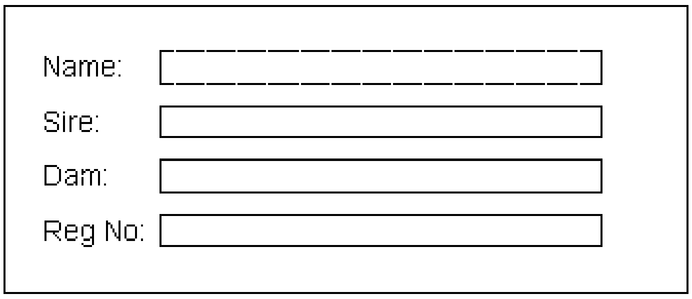
We could keep adding more fields and text as we did previously but there is a faster way. We can simply take what we have done and duplicate it.

In this example we will select the static text “Name:” and the field, the select **Edit-Duplicate** (or Ctrl-D). This will cause a copy of the selected items to appear 0.2” to the right and bottom of the original. The copy will be selected so click on it to move it to directly underneath. Now when we hit Ctrl+D again the second copy will appear with the same offset we have just set.

The next step is to change the static text and field values. To change the static text, select the text toll and click into the text and start typing. To change the fields simply select the fields one by one and using the field drop down list box change their values.

The result is shown in Figure 3.4.

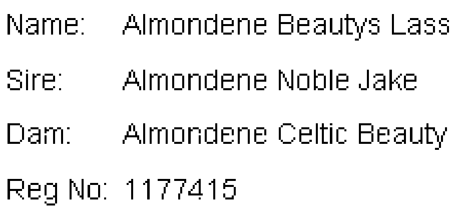
**Figure 3.4**



If the users were to click on the record, then “Pedigree” would appear in the edit box, indicating it is set to the Pedigree table. If the top field was clicked on then it would show “Name”. If the second field was clicked on it would show “Sire”..

After all that work it should appear as shown below in the forms view.

**Figure 3.5**

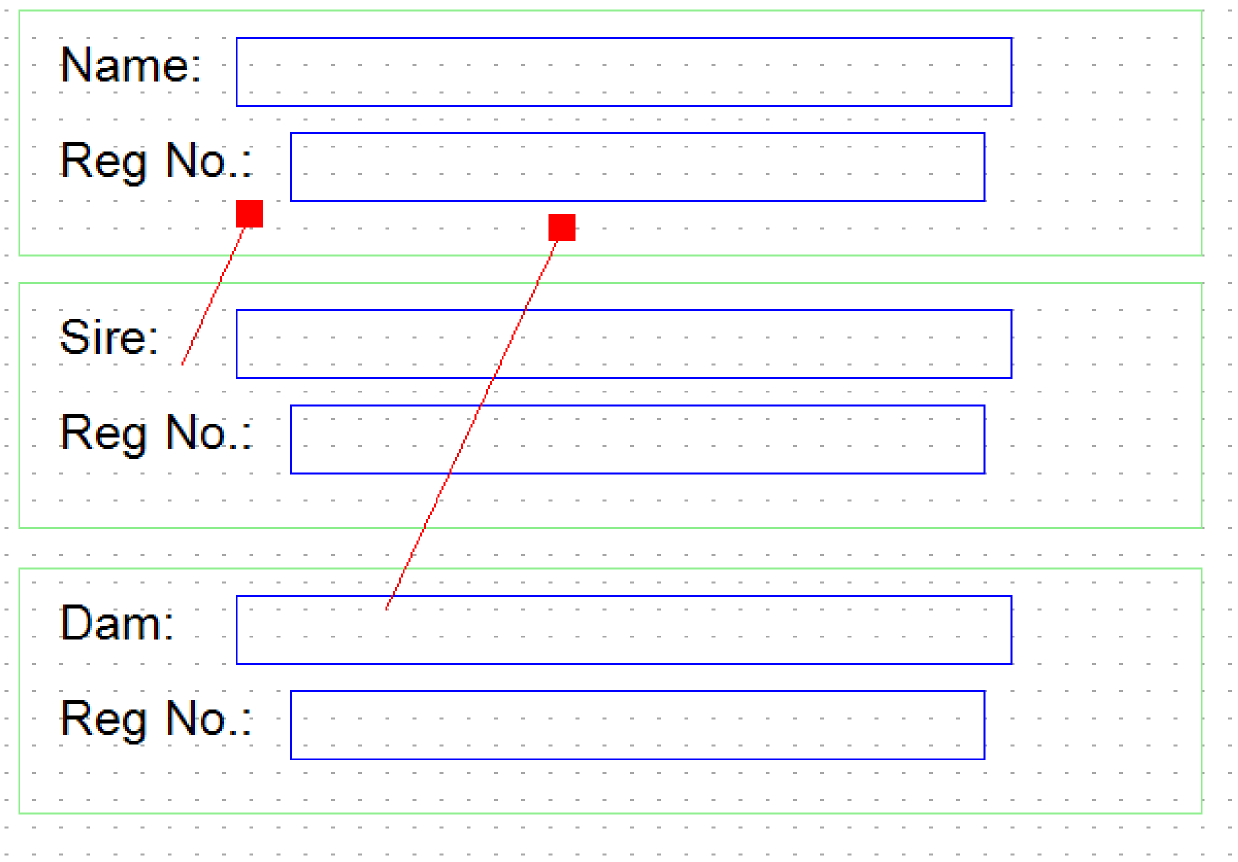

Name: Almondene Beautys Lass
Sire: Almondene Noble Jake
Dam: Almondene Celtic Beauty
Reg No: 1177415

## 2.6 Adding Links

The form we have created is "flat", ie it doesn't have data from any record except the current record. By using links we can create complex forms which show data related to the current record such as ancestors in a pedigree.

In this example our intention is to add a sire and dam record that will also show the registration number of the sire and dam.

**Figure 3.6**

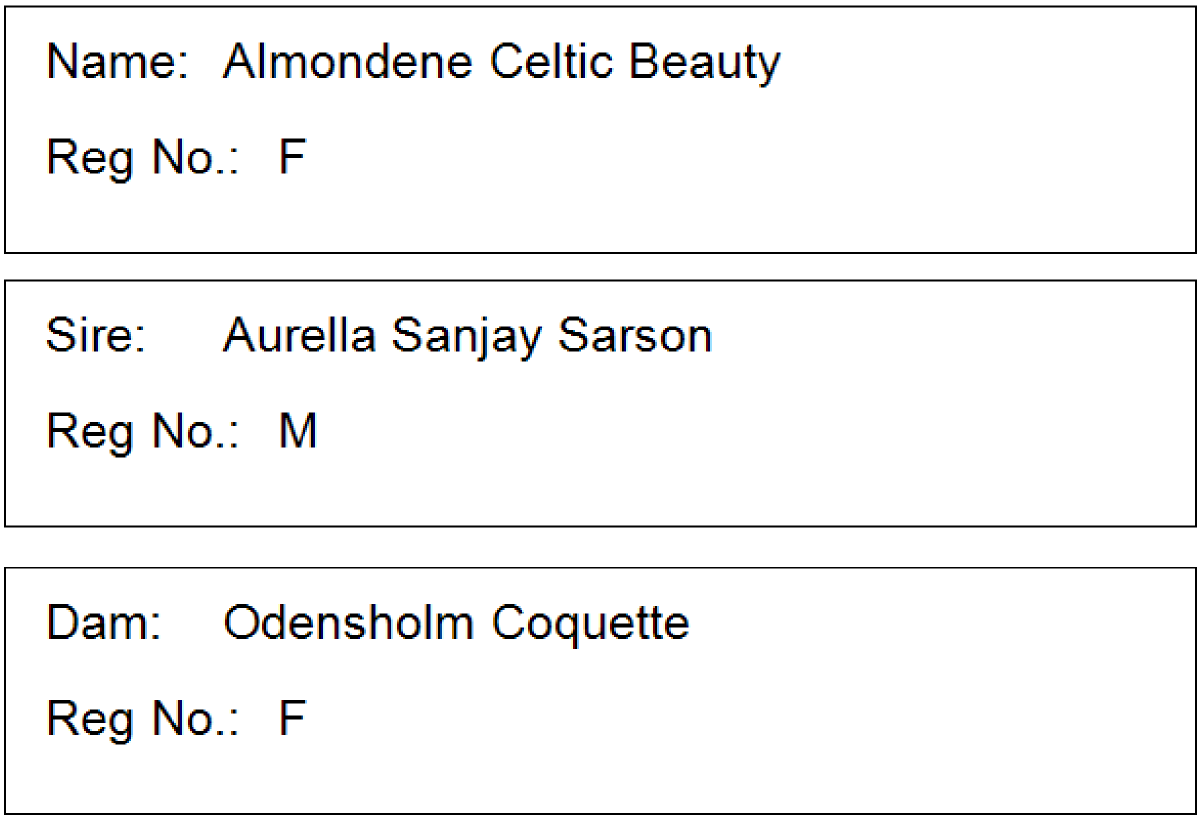


The steps for creating this new form are as follows:

- Delete the sire and dam text and fields - we won't need them.
- Move the Reg. No. text and field up below the name text and field.
- Readjust the size of the record to neatly enclose the two fields.
- Select all of this (use an area selection) then **Edit-Duplicate**.
- Adjust the position of the copy below the original then **Edit-Duplicate** again to produce a second copy. We now have three records (green rectangles) each with a Name and Reg No. field in them.
- Use the text tool to change the text in the 2nd and 3rd copies to Sire and Dam.
- Use the link tool to add two links and make sure that the blob ends of the links start in the top record (our root record) and the non blob ends finish in the two copied records.
- Using the select tool, select the first link (Leftmost) and change it to Sire via the field box drop down list, then change the 2nd link to Dam.

Your form is now complete and in forms mode will appear as show in Figure 3.7:

**Figure 3.7**



Name: Almondene Celtic Beauty

Reg No.: F

Sire: Aurella Sanjay Sarson

Reg No.: M

Dam: Odensholm Coquette

Reg No.: F

The techniques described above are sufficient to design 90% of forms but there are still some more advanced techniques which must be mastered for the more difficult forms.

## 2.7 Records inside Records

Some forms will require the designer to locate one record inside of another record. For example if we took the previous sample and wished to show the address of the owner as shown in design view (Figure 3.8) and forms view (Figure 3.9).

**Figure 3.8**

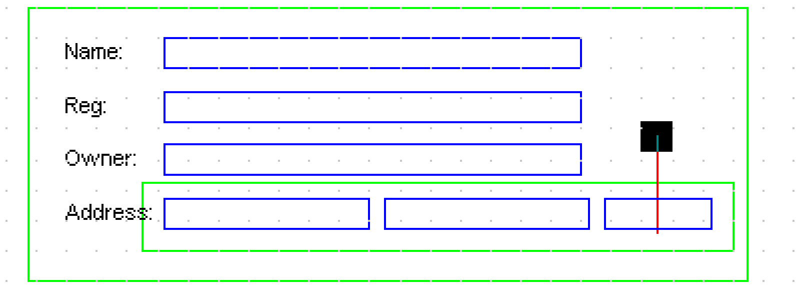
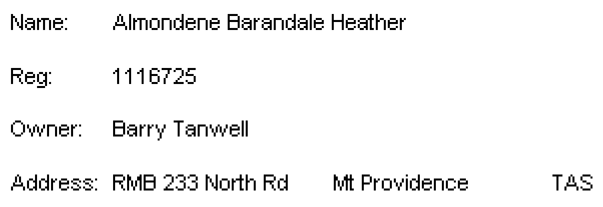


**Figure 3.9**



| | | | |
|---|---|---|---|
| Name: | Almondene Barandale Heather | | |
| Reg: | 1116725 | | |
| Owner: | Barry Tanwell | | |
| Address: | RMB 233 North Rd | Mt Providence | TAS |

The steps in designing this new form are as follows:

- Delete the sire and dam records from the previous sample.
- Extend the record to be large enough to include the owner and address fields.
- Select the Reg. No. text and field and duplicate them using **Edit-Duplicate** and position the duplicate below the original with sufficient vertical space.
- Create another duplicate. Note the second duplicate has the same spacing from its original as you set for the first.
- Change the first copy's text to Owner and field to Owned.
- Add a new record enclosing the last field and set it to Contacts table.
- Select the record and use **Send To Back**. This step is crucial, since when determining which record encloses a field, it is the record which is first in the display order (i.e. sent to the back) which will be chosen.
- Select the Owner record then select the field inside it and set it to "Street Address" via the field combo box.
- Resize that field and make two duplicates and set them to "Suburb" and "State" and position them horizontally.
- Add a link as shown and set it to Owner.

Your form is now ready to test by switching back to forms view. This example has shown the following important concepts:

- Display order of records affects which record a field and link belongs to
- Use of tables other than Pedigree table.

## 2.8 Using Expressions

The example above shows that an owner's address can be constructed by placing the three address fields, viz "Street Address", "Suburb" and "State" in a horizontal line. The main problem with this is that we have to fix the position of these fields, ie we can't just string the three fields together into one big field. Well actually you can and the way it is done is to use Expressions.

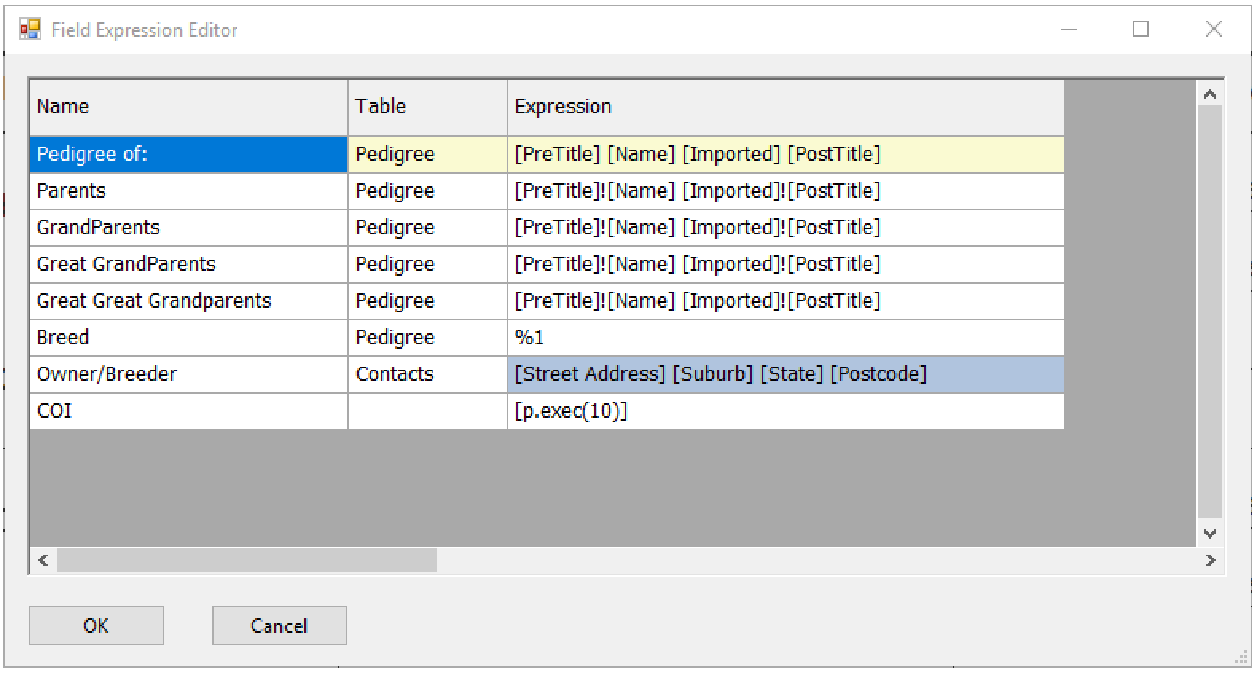
An expression is really just a compound field, ie it consists of one or more fields strung together.

To show the use of expressions in our example, follow these steps.

- Delete the "Suburb" and "State" fields then resize the "Street Address" field to extend the full length of the record.
- Next select the "Street Address" field and change it to "Expr1".

- We now have to define what is in Expression 1. Select **Edit-Expressions**…
- This will bring up a dialog box as shown in Figure 3.10.

**Figure 3.10**

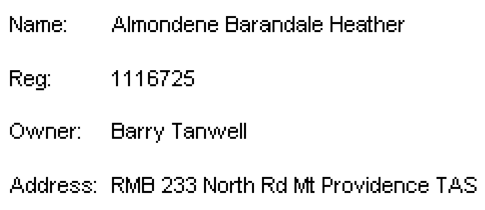


| Name | Table | Expression |
|---|---|---|
| Pedigree of: | Pedigree | [PreTitle] [Name] [Imported] [PostTitle] |
| Parents | Pedigree | [PreTitle]![Name] [Imported]![PostTitle] |
| GrandParents | Pedigree | [PreTitle]![Name] [Imported]![PostTitle] |
| Great GrandParents | Pedigree | [PreTitle]![Name] [Imported]![PostTitle] |
| Great Great Grandparents | Pedigree | [PreTitle]![Name] [Imported]![PostTitle] |
| Breed | Pedigree | %1 |
| Owner/Breeder | Contacts | [Street Address] [Suburb] [State] [Postcode] |
| COI | | [p.exec(10)] |

- First change the name of "Expr1" to "Address" then set the Table to "Contacts". Note this also causes the "Insert Field" combo to be loaded with the list of available fields from that table.
- Next select the "Street Address" field from the field combo. It will now appear in the Expression edit box enclosed in square brackets. Type a space character - it will appear after the "[Street Address]" in the Code edit box.
- Now select the "Suburb", type space, then select the "State" fields. The result is shown in Figure 3.11. Note the spaces between fields which you typed.
- Click the OK button.

The result will be as shown in Figure 3.12.

**Figure 3.12**



| | |
|---|---|
| Name: | Almondene Barandale Heather |
| Reg: | 1116725 |
| Owner: | Barry Tanwell |
| Address: | RMB 233 North Rd Mt Providence TAS |

# 3 Editing Forms

This section describes how to edit your own forms.

## 3.1 Selecting Objects

Entities can be selected as follows:

- Always ensure that the Select tool  is selected (it will appear pressed, i.e with a light blue background).
- Select a single entry by clicking on it with the mouse.
- Select a group of entities within an area by clicking at the top left of the area, dragging the mouse down to the bottom right of the area and releasing the mouse button.
- You can keep selecting more entries by pressing the Shift key down while performing either of the previous selects.
- Deselect entities in the same way as above, i.e. hold the shift key down when selecting - <u>if an entry is already selected it will be deselected</u>, if it is not selected then it will become selected.

Entities which are selected are displayed as have four handles or solid black squares, one in each corner.

## 3.2 Adding Entities

Any entity can be added by selecting the appropriate tool, e.g. text, line, rectangle as described in Section 5.

## 3.3 Moving Entities

First select the entities to be moved as described in Section 6.1 then click on any of the selected entities (but not in a blob) then drag the mouse to the desired position and release.

When moving entities ensure that the original relationships between the records, fields and links is maintained, i.e. that fields remain within their records and that links start and end within their original records.

## 3.4 Resizing Entities

Just as objects (like text or lines) can be moved, they can also be resized. In the case of text this <u>only changes the area in which they are bounded</u> it does <u>not</u> change their font size.

To resize, first select an object by clicking on it. Four ‘handles’ should appear as square black blobs. Click in one of the blobs and drag then release the mouse button when the correct size is achieved.

## 3.5 Deleting Objects

Any object can be deleted - note that it cannot then be recovered, except by either not saving the form file or if the file has been saved, then copying the forms file from a backup.

To delete an object or objects simply select them and then use **EDIT-Delete Selected**. This can be useful for removing fields from the standard pedigree forms if they are not required, eg remove registration numbers.

## 3.6 Changing Text

Simply select the text tool and click the cursor at the point at which you wish to add text or delete text. You can use the left and right cursor keys to move the insertion point. Use the backspace key to delete text.

To change the size or typeface of text, use the drop down list boxes for font and font size located in the design view's toolbar.

## 3.7 Changing Records

A record has only one value and that is the table it belongs to.

To change the table which the record belongs to first select it. Its table value will be shown in the Table drop down list box. Simply click on the list box and select a different table. Note you may need to scroll the list to see all available tables.

## 3.8 Changing Fields

- A field also has only one value and that is one of the available fields of the record it belongs to. For example, if a field was contained in a record which was set to the Entry table, then it could be set to any field in the table.

The available fields also include a list of expressions, vix “Expr1”, “Expr2” up to “Expr11”:

To change a field, go to design view then first select the enclosing record - this will enable the list of fields to be loaded. Next select the field by clicking on it and click on the list box to display the available fields, select the field.

If you set the field value to one of the table names then no more need be done. If however you set a field to one of the expressions then you will need to define the expression if it hasn't already been done as shown in the next section.

## 3.9 Defining Expressions

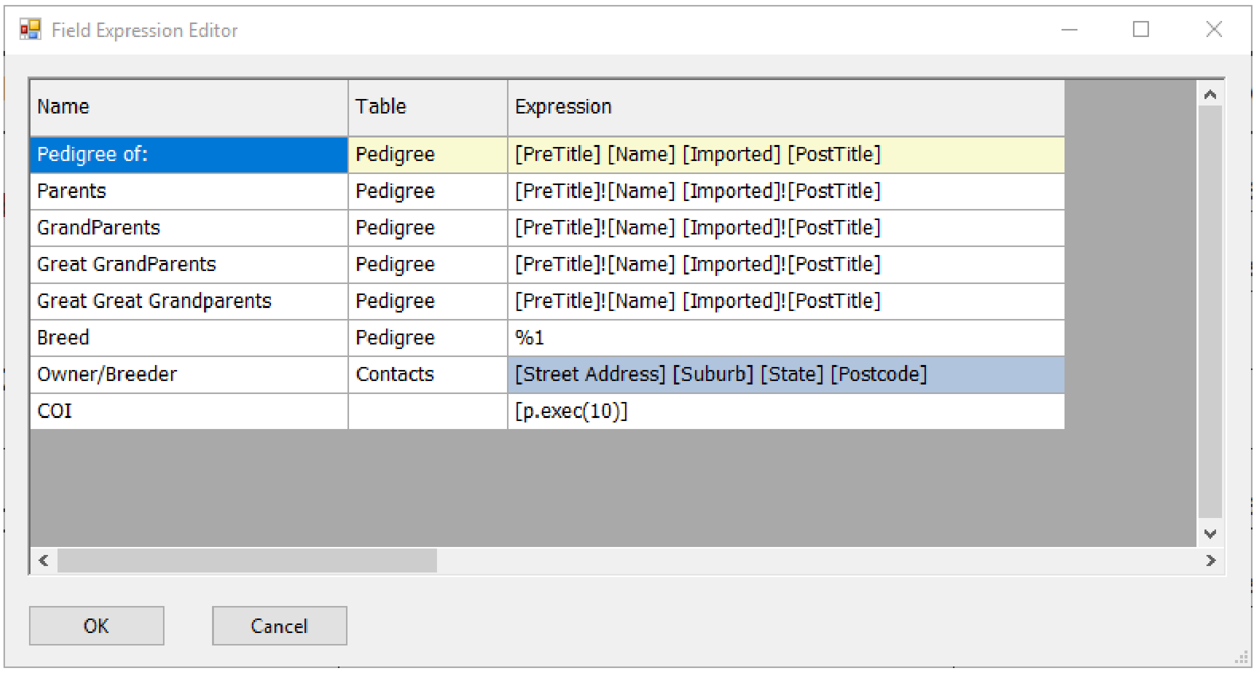
The fields used in designing forms can be either:

- a field value - eg “Name”, or “Colour” or “Sire”;
- an expression value, eg Expr1, Expr2

If a field refers to an expression then, when the form is being displayed, the field will be replaced by the expression. An expression is actually just a compound field, ie it consists of one or more fields strung together, eg the large compound field which appears at the top of most pedigrees is “title name imported obedience”.

Up to 10 field expressions are allowed per form and they are referred to as Expr1, Expr2… Expr10 – although the user can alter these names to something more descriptive like “Parents” – which implies that expression is used in the Parents fields. The ***Edit-Expressions*** command allows these Expressions to be edited or new ones to be created.

After selecting ***Edit-Expressions***, a dialog box appears as shown below.



| Name | Table | Expression |
|---|---|---|
| Pedigree of: | Pedigree | [PreTitle] [Name] [Imported] [PostTitle] |
| Parents | Pedigree | [PreTitle]![Name] [Imported]![PostTitle] |
| GrandParents | Pedigree | [PreTitle]![Name] [Imported]![PostTitle] |
| Great GrandParents | Pedigree | [PreTitle]![Name] [Imported]![PostTitle] |
| Great Great Grandparents | Pedigree | [PreTitle]![Name] [Imported]![PostTitle] |
| Breed | Pedigree | %1 |
| Owner/Breeder | Contacts | [Street Address] [Suburb] [State] [Postcode] |
| COI | | [p.exec(10)] |

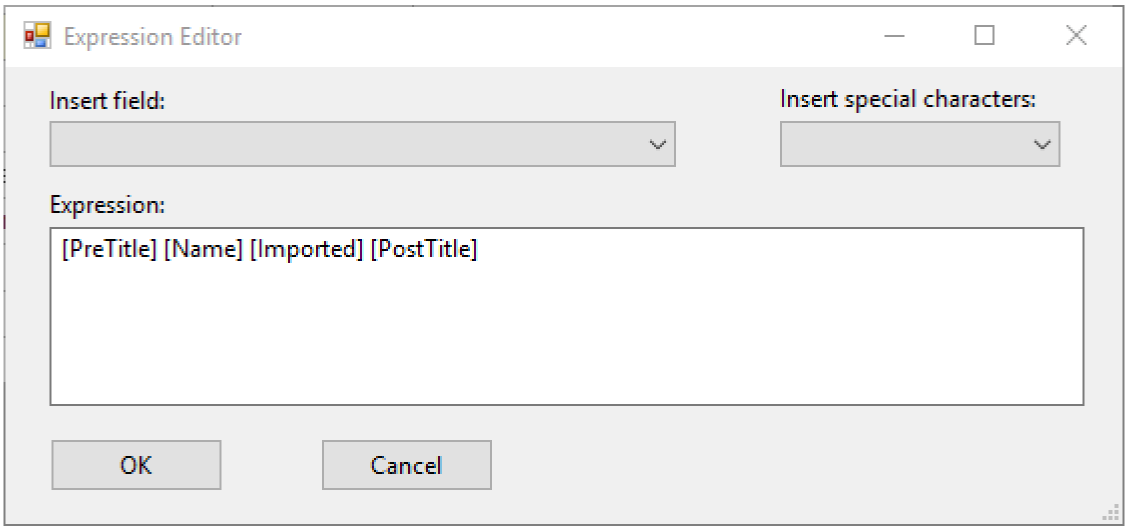
Firstly determine the expression you wish to edit by clicking on the Expression cell that corresponds to the index or where the Comment field indicates it applies to the fields of interest.

Usually you won't need to change the table if its an existing expression, but if you are creating a new expression then select the table the expression belongs to. Now all that remains is to enter the expression in the field labelled "Expression".

Any characters you type in the edit box will appear as is in the form except for the following special characters (which can be inserted via the "Insert Special Character" drop down):

- The '!' will be replaced by a new line.
- The '%' character followed by a single digit refers to a calculated field. There are only two calculated fields, viz 0 is for percentage of an ancestor in the final dog, and 1 is for the Breed field. Hence "%0" in the expression would get replaced with the percentage.
- The '[' character followed by a field name and then a closing ']' means "repalace this with the contents of this field". Hence is "[name]" wa in the expression then it would be replaced with the value from the Name field.

A fast way of putting in fields (ie those enclosed in '[' and ']') is to click on the drop arrow in the edit box. This will bring up another dialog as shown below:



[PreTitle] [Name] [Imported] [PostTitle]

To insert a field in the expression select the field from the field combo box. Every time you select a field it will appear in the expression. Click on OK to return to the previous dialog.

## 3.10 Changing Links

A link is set identically to a field. Each end of the link must also be contained within a record. The record which controls which values a link may have is the one which contains the source end of the link (has a large rectangular blob on it).

Links may have the values as listed above for fields except for the special calculated values. In addition though it has two possible special values as listed below:

- “Expr1”. When this value is selected, then the linked records are merely copies of each other. This is used in some of the show entry forms to extend a record to isolated islands of fields.
- “Expr2”. When this value is selected then the destination record is duplicated for every record in the table. It is a very special case that is used only for the Litter Records form.

Note while the two special links listed above are reffered to be names as Expr1 and Exp2, they have no relationship at all with expressions -they’re just convenient names!

## 3.11 Units of Measurement

The base unit of measurement used in forms is 0.01” (one tenth of an inch). Normally the Snap To Grid option is set (it has a tick next to it) on the VIEW menu, which means that the grid is 0.1”. To use a 0.01” grid, simply unset that option.

When moving entities or resizing, the status bar will show either position or dimension.

## 3.12 Aligning Entities

Aligning entities is the process of moving them so that either their, left, right, top or bottom extents are the same or their height or width are the same.

The toolbar actually contains buttons to perform all these alignment functions. First select the entities to be aligned then the relevant button. Note by hovering the cursor above the button its command function will be displayed in a “tool tip”

# 4 Worked Examples

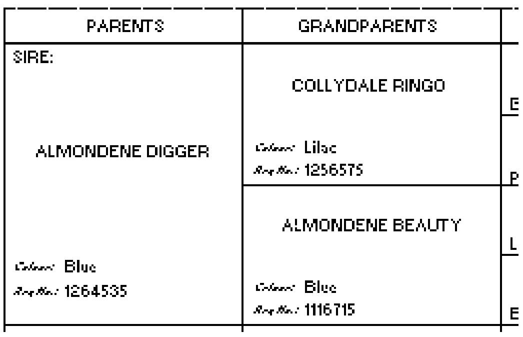
This section describes some simple modifications that could be performed on any of the standard forms.

You may wish to save any changes to forms in a different form file and thus leave your standard form files in tact. Use the **File-Save Forms File As** command to make a copy of the current form file and save the file with a different name, eg BREED.FRM.

## 4.1 Deleting Unwanted Fields

Some of the standard pedigrees may be just what you want except they may have some fields that you don't need. As an example take the "4G Pedigree (A4 Fancy)". It has colour in the parents, grandparents and great grandparents and apart from this unwanted field is perfect for the user's requirements. A sample of one part of the pedigree is shown in Figure 5.1.

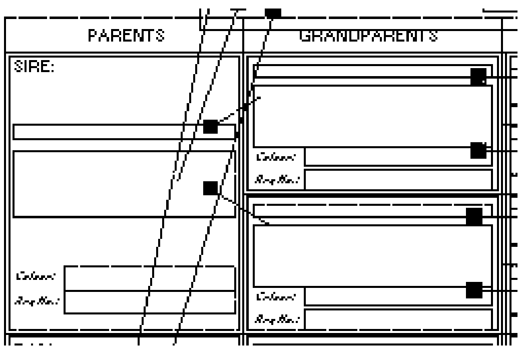
**Figure 5.1**



| PARENTS | GRANDPARENTS |
|---|---|
| SIRE: ALMONDENE DIGGER Colour: Blue Reg No: 1264535 | COLLYDALE RINGO Colour: Lilac Reg No: 1256575 |
| | ALMONDENE BEAUTY Colour: Blue Reg No: 1116715 |

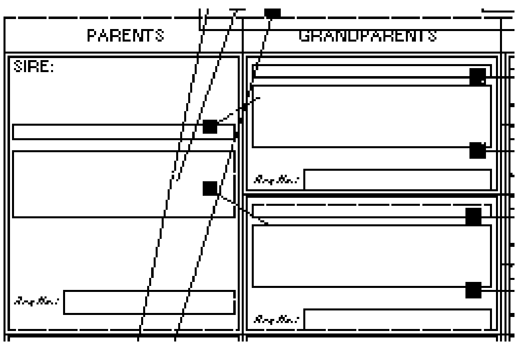
To change this, first enter edit mode, and the pedigree will change as shown below. Then just select the text "Colour:" wherever it appears holding the shift key down after selecting the first entity so that the selections keep accumulating. All of the "Color:" text will now be selected so just hit the Delete key to remove them.

**Figure 5.2**



The only items to remove now are the fields. As for color, select the first color field and, holding the shift key down, keep selecting the others, then hit Delete. The resultant form should appear as shown in Figure 5.3.

**Figure 5.3**

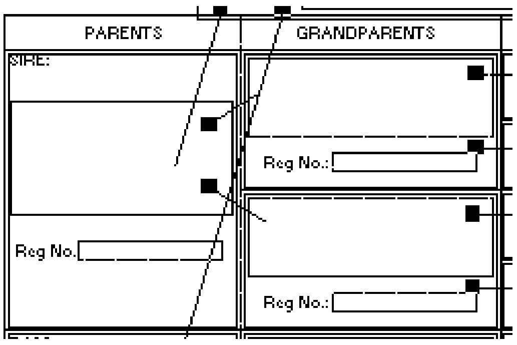


Exit design view and view the results. Remember to Save Form, otherwise your changes will not be permanent.

## 4.2 Adding Extra Fields

Let us assume that the 4G Pedigree is just what you want except that it doesn't show hip scores. Again we will work through the changes required showing just a portion of the pedigree as shown in Figure 5.4.

**Figure 5.4**



Rather than creating the text and field from scratch it is always easiest to copy one that is similar. So first step is to select the "Reg. No." text and the field to the right of it. Next do **Edit-Copy**, then **Edit-Paste**. The copy which was pasted will appear selected. Click on it and drag it so that it appears positioned directly below the original as shown below.

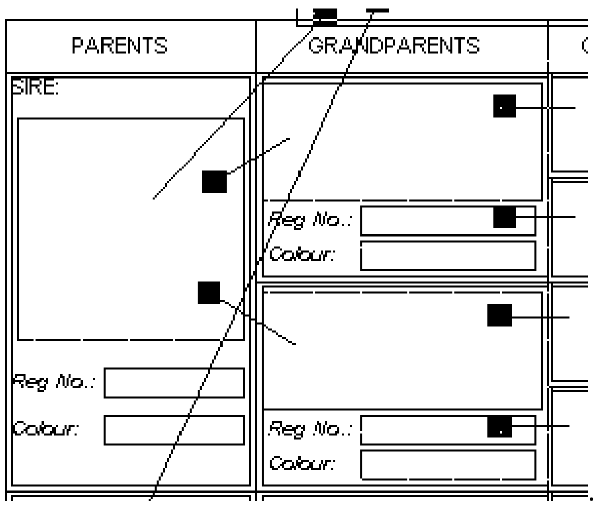
## 4.3 Changing Fields

Some pedigrees may be close to what is desired except that a field may need to be changed. As an example take the 5G A4 Pedigree. It has colour and Reg. No. in the first two generations. Lets say we wish to change colour to hip score. Just follow these steps.

Select the text tool and click in the static text "Colour". Use the arrow keys to navigate to the end of the text and hit backspace to delete all of it then type in "Hip Score:", then select the field next to it. Its field will appear in the edit box as "Colour" - hit the drop down button and select the "Hip Score" field then hit Enter.

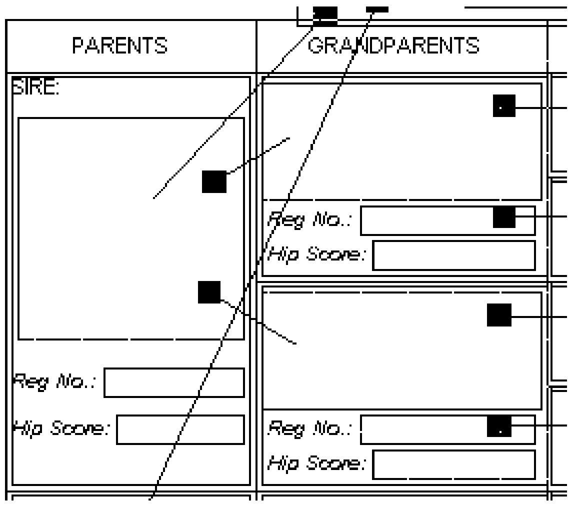
So far we have changed just one copy of this. Rather than repeat this for all fields and text, we will just select all of the other text, then delete it. Next we will select the text and field, hit *Edit-Copy* to copy this pair to the clipboard then we paste copies to replace the ones we deleted.

**Figure 5.5**



The result of these changes will be as shown in Figure 5.6:

**Figure 5.6**



## 4.4 Copying Forms

There may be situations where the user wants to keep a form but also use a modified version of that form. This can be accomplished as follows:

- Select ***Form-New***. A dialog box will appear. Enter the name of your new form and hit OK.
- Select the form you wish to copy from and select all the entities in that form by using ***Edit-Select All***.
- Copy the entities to the clipboard using *Edit-Copy* and view your new form by selecting it from the Forms menu.
- Click the mouse in the top left corner (this establishes where the top left corner of the pasted copy will appear). Paste the copy in using *Edit-Copy*.
- Adjust the position of the forms entities by clicking on any one of the selected entities and dragging all the entities to the correct position.

# 5 Text Reports

V6.4 of PedPub has a completely re-written text report module. Text reports written for previous version of PedPub will not work in V6.4 or later and there is no conversion available.

Text reports now are written in an advanced scripting language called RIO.

## 5.1 Introduction

Breedmate is one of the few (possibly only) pedigree programs with its own scripting language. The language is called RIO. Currently it is only accessible as a text report which means that you will see them listed under the Forms-Reports section in the Chooser bar on the left of the Breedmate window.

In simple terms you can use RIO to write your own analysis functions that either produce text outputs (which incidentally can be easily written to output to say Excel) or will mark records so they can be filtered in the main grid view.

### 5.1.1 Hello World

The first sample generally listed for any language is “Hello world!”. Here is the program:

```
Hello world!
```

That’s it. Just put in the text. (This is possibly the shortest Hello World program ever). Any normal text is just output. Apart from normal text there is only one other type of text – it’s called program text. All program text starts with the two character sequence <# and ends with #>.

Another example of the Hello world! Program is as follows:

```
<#
println 'Hello world!'
#>
```

It’s a one line program (not counting the delimiters that indicate this is program text). The program has two parts: a println command which prints anything to the right of it and adds a new line after it. The second part is the string “Hello world!” – note that all strings are wrapped in double quotes.

As a slight variation on the above we can also do:

```
<#
println 'Hello', 'world!'
#>
```

In the above case we have two strings separated by a comma. RIO simply prints any list of arguments in order. Println will put a space automatically between arguments if the previous argument is not blank.

### 5.1.2 Using a variable

In this next variation we will assign the string to a variable then print it:

```
<#
$var = 'Hello'
```

```
println $var
#>
```

Note all variables start with $.

### 5.1.3 Simple maths and a loop

Combining the above with some maths and a while loop we have the following short program that prints the numbers from 1 to 10 and their square.

```
<#
$var = 1
while $var < 10 then
    $sqr = $var * $var
    println $var, $sqr
    $var = $var + 1
end while
```

The output from this is:

```
1, 1
2, 4
3, 9
4, 16
5, 25
6, 36
7, 49
8, 64
9, 81
```

The main elements of the while statement are while TEST then do something end while. But we could also do the above with a for loop in which case it looks like:

```
<#
for $i in 1:10
    println $i , $i *$i
end for
#>
```

This is interesting but Breedmate is after all a pedigree program designed to deal with database tables, records and fields so we'll now start exploring what commands are available to handle this type of data.

### 5.1.4 Print the name of the first record

Starting with this simple example that simple prints the name from the first record of the Pedigree table. BTW to see the first record of a table you need to click the unsort button on the grid view toolbar because normally it is in alphabetic order on the first column.

```
<#
$var = $Pedigree[0]
println $var[0]
#>
```

In the above we have used the pre-defined $Pedigree variable (a pre-defined variable exists for every table in your database and is automatically created for you). Then we index it with 0 to get the first field which incidentally is the name field.

Alternatively we could have written it explicitly like this

```
<#
$var = $Pedigree[0]
println $var[“Name”]
#>
```

Lets say we want to print the name and reg number separated by >> then we would write:

```
<#
$var = $Pedigree[0]
println $var['Name'], ' >> ', $var['Reg No.']
#>
```

Note that println above has three comma separated arguments.

### 5.1.5 List the name and registration of all champions

Now for something more useful. The following code lists all champions including their name and registration #.

```
<#
println 'Test to print name and reg no of all champions'
for $p in $Pedigree where $p['Titles'].like('CH')
    println $p['Name'], $p['Reg No.']
end for
#>
```

In this example $p is a bookmark (i.e. a reference to a record in a table). $Pedigree is the table so when we say $p in $Pedigree RIO will loop over every record in that table. The for statement has a where clause that tests to see if the Titles field contains the string “CH” anywhere in it regardless of case. Inside the for loop we have just one println statement to print name and reg #. Note that the square brackets are called an “indexer”. The bookmark represents a record which can be considered a list of fields, The indexer allows access to one of those fields either by position or by name. Note that the name is case sensitive and must match exactly.

Here is part of the output of that script:

```
Test to print name and reg no of all champions:
Almondene Bärandalë Heather, 1116725
Almondene O'Barandale Lassie, 1116675
Almondene Beautys Lass, 1177415
Almondene Celtic Beauty, 1163625
Almondene Charger, N1264555
Almondene O'Dell, 14702
Almondene Digger, 1264535>
```

The above works quite well but can be SLOW for large database tables because it is having to read all the records in the pedigree table then filter them based on whether the "PreTitle" field contains "CH". The faster way is to get the database to do the filtering using an SQL query. This is done using the "query" function. So rewriting the above we have the following.

```
<# println 'Test to print name and reg no of all champions'
   for $p in query("select * from pedigree where Pretitle like '%ch%' ")
   println $p['Name'], $p['Registration']
   end for
   #>
```

Note in the above, in SQL, field and table names are case insensitive so we could "select * from PEDIGREE" or "select * from pedigreEE". Also the "like" statement also does case insensitive matching. The percent % either side of ch tells "like" to match "ch" anywhere within a string so it will match "US CH" or "CH Grand". The "query" function is covered in more detail in the "Scripting Reference"

For those unfamiliar with SQL the * in the "select * from" means return ALL the fields in the table. We could possiby make our query slightly more efficient, and therefore run faster, by only fetching the fields we want to use. Son in this case instead of "select * from pedigree where Pretitle like '%ch%' " we could make it "select name, registration from pedigree where Pretitle like '%ch%' "

If we are printing lots of fields out, then instead of using "println" with lots a $p['Name'] we could instead use the "expr" function with an expression. Here is what this would look like.

```
<# println 'Test to print name and reg no of all champions'
   for $p in query("select * from pedigree where Pretitle like '%ch%' ")
   println expr($p, "[Name] [Registration]")
   end for
   #>
```

### 5.1.6 Mark all champions

In this sample, rather than print the champions, we mark them so that they can be viewed or filtered in Grid view.

```
<#
println 'Mark all champions'
$count = 0
for $p in
    $Pedigree where $p['Titles'].like('CH')
    $p.setMark(1)
    $count = $count + 1
end for
println 'There were', $count, 'champions marked.'
#>
```

The report is very similar to the previous report but inside the for loop we call the “setMark” method of the record variable with a value of 1. 1 corresponds to the red mark. If we had used 2 then the mark would be orange, 3 is yellow and so on. We also keep a count of how many we have marked and print that out at the end.

### 5.1.7 Print a record of a specific name

Indexers are very powerful functions. As well as working on a bookmark to get a field they can also work on a table to get a bookmark (i.e. a record or row). In the following example we print out every field from a record with a specific name:

List record by name

```
<#
$var = $Pedigree['Almondene Beauty']
println $var
#>
```

The result of running that on the DogSample.bmx database is:

List record by name

```
Almondene Beauty, F, , Collydale Toss, Almondene Meg, 1116715, , , Blue, , Barry
Tanwell,
Barry Tanwell, , , , , TAS 294, , , , , , , , , AKC, , , rhapsody1.jpg, , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , 2011-07-28 19:35:04, , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , ,
```

### 5.1.8 List all entries bred by this breeder

In this example we will list the names of all entries that were bred by the breeder of the currently selected record.

```
<#
$breeder = $Subject['Breeder']
println 'Entries bred by this breeder: ', $breeder
for $p in $Pedigree where $p['Breeder'] == $breeder
println $p['Name']
end for
#>
```

In the first line we use the predefined variable $Subject – this is the currently selected row (i.e. record or bookmark) in the currently selected table. We then get the contents of the Breeder field and assign it to the $breeder variable (this is just a convenient name – we could have called this variable anything like $fred – the important thing is it starts with $).

In the second line the println statement will just output the fixed text starting with “Entries owned…” and append the $breeder to the end. We output this first so its clear what entries we are matching to.

The third line is a for statement which iterates (loops over all values) over the Pedigree table basically looping over all the record. Each time through the loop it assigns the record to the $p variable. The for statement has a where clause that tests if this record is what we want by pulling out both the Breeder and Owner fields and comparing with our nominated breeder. If either is a match then the for loop will use that record.

Inside the for loop we have just one statement – a println which uses the $p variable – which in this case contains a record, and then gets the field in that record called Name and prints it.

The result of running the above on the DogSample.bmx database is:

```
Entries owned or bred by this breeder: Barry Tanwell
Almondene Bärandalë Heather
Almondene O'Barandale Lassie
Almondene Beauty
Almondene Black Beauty
Almondene Gallant Lad>
```

### 5.1.9 Mark all champions

If all you want to do is filter the records you see in grid view then this can be done by using the setMark() command as we do in the following example which is a simple modification of the “List all champions” sample.

```
<#
```

```
println
'Mark all champions'
$count = 0
for $p in $Pedigree where $p['Titles'].like('CH')
    $p.setMark(1)
    $count = $count + 1
end for
messageBox('There were '+ $count+ ' champions marked.')
#>
```

Inside the for loop we use $p (which is the current record or row in the table) and call setMark(1) which sets the marks on this record to 1 (the red mark). We also keep a count of how many records we have marked and display a message using the messageBox() command which takes a string argument.

Note there were the argument to the messageBox command was three values concatenated or joined together. Two of them are strings while the middle is an integer (or whole number). RIO automatically converts the number to a string before doing the join.

### 5.1.10 List all entries born within five years

This is an example of handling dates. The first line merely print the subjects (currently selected record in the grid view) name and date of birth.

We then read the DOB and put it into $date. Dates are internally stored as numbers and each daya is equivalent to 1 so adding say 7 is the same as adding a week. By adding 365 * 5 we have added five years to the date and assigned it to $datePlusFive.

We go into a loop over all the records in the Pedigree table testing if the DOB field is after our subject's DOB and before the DOB plus five years. Inside the loop we print the records name and DOB.

```
<#
println 'Mark entries born within five years of this record: ' + $Subject['Name'] +
' who was born: ' + $Subject['DOB']
$date = $Subject['DOB']
$datePlusFive = $date + 365.0 *5// add five years to the subject's DOB
for $p in $Pedigree where $p['DOB'] < $datePlusFive and $p['DOB'] > $date
    println $p['Name'] + ', ' + $p['DOB']
end for
#>
```

Partial output is below:

```
Mark entries born within five years of this record:
24kt-kats Buddy who was born: 04-10-1996
A'far Major Paine, 26-01-1997
Abizaq Southern Exposure, 10-03-1998
Abyfinns Hermione, 20-11-2000
Abyriginal Blonde Ambition, 02-11-1996
Abyriginal Desert Hawke, 02-11-1996
Abyroad Moonlight Sonata, 20-04>
```

## 5.2 RIO Scripting Reference

All text in a RIO program is either plain text or program text. Plain text is just text not enclosed in <# and #> while program text is enclosed in it.

### 5.2.1 Definitions

RIO programs consist of statements like **if**, **while**, **for** and uses **variables**, **operators** and **constants** combined into **expressions**. These are all defined in the following sections.

### 5.2.2 Variables

Variables are used for storing values. All variables must start with the dollar sign $. The characters after the $ must be alphabetic or numeric, e.g $Hello, $My9hole. The names are case sensitive so $Fred is different from $fred.

The types of values that can be stored in a variable are as follows:

- Integer – these are whole numbers like 1, 9876, -55
- Double – these are floating point number like 77.3456
- Date – e.g. getDate("12-22-2010")
- String – like "This is a string" – basically a series of characters wrapped in double quotes.
- List – simple a list of values
- Bookmark – this is a unique reference to a record in a particular table
- Table – this is a reference to a database table

### 5.2.3 Operators

These are symbols that are used to perform operations on variables, e.g add or multiply. A complete list of operators is below:

- + - used to add to integers or numbers together, also to concatenate strings, i.e. join them end to end
- - multiply two integers or doubles, e.g. 3 * 4 = 12. If either argument is a double then the result is a double.
- / - divide two integers or doubles, e.g. 8 / 2 = 4.
- >, < , <=, >=, == , != these are respectively, greater than, less than, less than or equal to, greater than or equal to, equals, not equals
- and – logical combine to values both must be true for the result to be true
- or - logical combine to values either can be true for the result to be true
- : - the format operator. The left side of the format operator is the value to be formatted. The right hand side is the format string. E.g. $Subject["DOB"]:"MM-dd-yyyy".
- .. - the range operator. The format is first value:last value, e.g 10:100 is all the numbers from 10 to 100 inclusive. The range can optionally also have a step, e.g. 1:10:2 means step from 1 to 10 in increments of 2.
- , - the comma joins values into lists so e.g. 2,3 is a list consisting of two values
- [index] – this is the indexer and it operates on the argument to its left to fetch one element from it either by position or name. The way it works depends on what is to the left of it. If the left argument is a table then an integer index will return the Nth bookmark (record) in the table while a string index will return the record of that name (i.e. the first column of the table). If the left argument is a bookmark then an integer index will return the Nth field (Name is the

0th, Sex is the 1st, DOB is the3 2nd) and if the index is a string it will return the field of that name from the record so $p[“DOB”] will return the date of birth field for that record.

## 5.3 Functions

There are no user definable functions in RIO (yet) but the following predefined functions are available:

- **pedigree**, **ancestors**, **reversePedigree**, **htmlPedigree** – all these functions take two arguments: a bookmark, number of generations and an optional string field expression
- **relations**, **fullSiblings**, **sireSiblings**, **damSiblings**, **damLine**, **sireLine**, **offspring** – all these functions take a bookmark followed by an optional field expression.
- **messageBox** – takes a string argument which it prints in a popup dialog box with an OK button. It is useful for giving messages e.g. like how many records your script marked.
- **getDate** – takes a string argument in your current date format (this will be what you see in say your DOB field if it is set to automatic – which it is by default). As an example in the US you could pass the following string in getDate(“11-27-2011”).

### 5.3.1 Query

The query function provides a fast way of running some reports. Basically any report where there is a statement like the following will be slow.

```
for $p in $Pedigree where $p["Sex"] == "M"
```

The above is going to be slow because it is reading every single record from the database table, THEN filtering it. It is much faster to run a database query which does the filtering itself as shown below.

```
for $p in query("select name, sex from pedigree where sex='M'")
```

Another example is the following report which lists all the entries bred by the Breeder in the current record.

```
[#
$breeder = $Subject["Breeder"]
println "Entries bred by this breeder: ", $breeder
if $breeder != "" then
for $p in $Pedigree where $p["Breeder"] == $breeder
println $p["PreTitles"], $p["Name"], $p["PostTitle"]
end
end
#]
```

The faster method using the query function is as follows.

```
[#
$breeder = $Subject["Breeder"]
println "Entries bred by this breeder: ", $breeder
for $p in query("select * from pedigree where breeder like '@breeder' order by
name")
println $p["PreTitle"], $p["Name"], $p["PostTitle"]// this version just prints
names and titles!
end
#]
```

A few technical points about the query command:

- Any valid SQL statement is allowed. Basic SQL is similar for both SQLite, which is what is used for desktop .DB files, and for MySQL servers. However, if you use SQL commands or syntax specific to SQLite or MySQL then the report will only work for that database.
- Note that @breeder in the query string gets replaced with the value of that variable and if that value is a string then the string is "escaped". Escaping is required for strings because strings are always delimited with single quotes, therefore the strings themselves can't have single quotes within them.
- If the variable has a datetime value then it is formatted as 'yyyy-MM-dd HH:mm:ss' which is the standard format for datetime in SQLite and MySQL. Note that it adds the single quotes areound the value whereas for strings the single quotes are not added so you need to add them yourself. The reason for this is that the string value might be part of a "like" statement which is used when you want to match anywhere in a string.

## 5.4 Methods

Methods are similar to functions but they occur after an argument and a dot. For example:

“Hello there!”.pad(5)

Would produce the result:

Hello

The method in this case is “pad”.

The currently supported methods are:

arg.toLower()– converts the string to all lower case so “Hello There”.toLower(0 will produce “hello there”

arg.toUpper() – converts the string to all upper case

arg.like(text) – returns true if arg contains the string text anywhere within it regardless of case

arg.pad(n) – arg must be a string and n is an integer. This method modifies arg to be a string padded out with spaces on the right to n characters. If the string is longer than n then it is truncated to n.

arg.setMark(n) – arg must be a record and n an integer. The mark on this record is set to n where n is a bit mask so setting 1 sets the primary mark red, setting two sets orange but setting three sets both red and orange.

### 5.4.1 Statements

The following statements are supported:

### 5.4.2 For statement

The format is:

for *VARIABLE1* in *VARIABLE2 where CONDITION*

*do something*

end for

The “where CONDITION” is optional.

### 5.4.3 While statement

The format is:

while *EXPRESSION* then

*do something*

end while

EXPRESSION can be anything which can be evaluated as true or false.

### 5.4.4 If statement

The format is:

if *EXPRESSION* then

*do something*

else

do something else

end if

EXPRESSION can be anything which can be evaluated as true or false. The else part is optional

### 5.4.5 Print statement

The print statement just prints the comma separated arguments to its right. The println statement is similar except it appends a new line character to the end. Print and Println will put a space automatically between arguments if the previous argument is not blank.

## 5.5 Miscellaneous

### 5.5.1 Expressions

Expressions are the same as described elsewhere in the user manuals. They are simple strings which contain either normal text or references to fields in a record. The references are the name of the field inside square brackets. If text is NOT inside square brackets then it is output as is. In the example below the println statement includes a call to expr() which returns a string which is the $expr text with the field references replaced by the value of those fields in the $Subject record.

```
$expr1 = "[PreTitle] [Name] [PostTitle] [Registration]"
println "Pedigree of: ", expr($Subject, $expr1)
```