Dexel Form user manual

Contents

What is Dexel Form?

Dexel Form is an Excel Add-in that provides a form based approach for the input and update of data records in a table in Excel. No customisation or programming is needed to use it on any table in Excel. It is simple and fast to use. And with a minimum of effort it can help you make your Excel tables much more robust in terms of data quality.

Dexel Form provides:

1. form-based creation, editing and filtering of records in a Excel table
2. built-in data validations
   a. data type validation (numeric, date, time, interval)
   b. uniqueness of values, or collections of values (composite keys)
   c. ensuring required fields are filled
   d. verification of data against related tables (lookups and conditional lookups)
   e. maximum and minimum length requirements for text input
3. custom data validations
   a. if you have defined validation formulae using Excel's built in Data Validation mechanism, then Dexel Form will enforce those validations
   b. Dexel Form also provides a more user-friendly mechanism for defining data validations using Excel formulae
4. lookups and conditional lookups for retrieval of data from related tables
5. definition of default values for new records. These can be either static (providing the same default value every time) or dynamic (based on a Excel formula). A particularly useful option is the ability to duplicate data from the previous record.
6. data protection. You can protect tables (either with or without a password) such that the table data can only be updated using Dexel Form. You can also limit the actions on the data in a table (e.g. do not allow records to be deleted).
7. multi-user access. With Dexel Sharing you can share a workbook such that multiple concurrent users can update the workbook at the same time, all the while maintaining data integrity and applying data validations.
8. multiline input of lengthy text fields, including text fields containing embedded new line characters (e.g. postal addresses).
9. an audit trail of changes to data.
10. data verification to highlight data errors in data imported from external sources.
11. programmability. If you want to use Dexel Form to manage the data entry dialog for hidden sheets, you can write simple macros to do so.

Prerequisites and limitations

Dexel Form is a Excel Addin.   It consists of a single file called **Dexel Form.xlam** which needs to be installed following the installation instructions below.

It has been developed entirely in Visual Basic for Applications (VBA) on Office 365. It may run on previous versions of Excel but this has not been tested.

Dexel Form does not use any Windows APIs and it runs fine on Excel on a Mac.

It supports up to 1,000 columns in a table. The number of rows is limited only by what Excel supports.

Dexel Form works only on desktop versions of Excel. It will not work on web versions of Excel.

Dexel Form will not work on workbooks that have been shared via Excel workbook sharing or co-authoring. However via Dexel Sharing it is possible to have multiple users work on the same workbook.

Installation

Requirements

For you to be able to install Dexel Form, you will need to have access to download and install an Addin. If your organisation's policies prohibit this, then you will not be able to install Dexel Form. Depending on the security settings, you may also need to explicitly Trust the Addin.

Installing Dexel Form involves copying the file **Dexel Form.xlam** to your Microsoft Excel Addins folder and then telling Excel to use it as an Addin. This is a straightforward process once you know where to look. The difficulties that some people encounter when installing addins are:

1. Your Microsoft Excel Addins folder is in a relatively obscure place:

   On Windows it is at **C:\Users\*username*\AppData\Roaming\Microsoft\AddIns**

   On Mac it is at **/Users/*username*/Library/Group Containers/UBF8T346G9.Office/User Content.localized/Add-Ins.localized/**

2. The dialog for activating Addins is also in a relatively obscure place and is not by default readily available in Excel.

In order to help mitigate the above difficulties, an installation helper script is provided.

Download and install

Download Dexel Form as a zip file from http://dexelform.com/download/

The zip file containing 3 files:

- **Dexel Form.xlam** which is the Dexel Form Addin
- **InstallDexelForm.xlsm** which is a helper script to help you install Dexel Form
- The **Dexel Form User Manual** (this document)

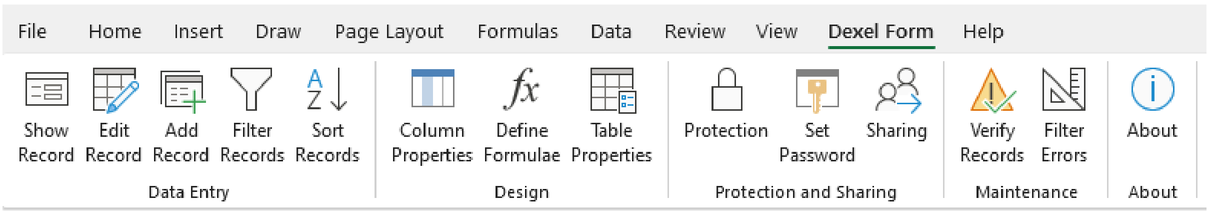
Zip files can be deceptive. When you look at them in Windows Explorer (or Finder on the Mac), they almost look like normal folders. However, they are not. The contents are only extracted on demand, so the install script will probably not work if you try to run it directly from your downloads folder. For this reason, copy the two files **Dexel Form.xlam** and **InstallDexelForm.xlsm** to your desktop (or some other location on your computer) and run the install script from there. You will be able to delete these files when the install is complete.

Once you have copied the files to your desktop, open **InstallDexelForm.xlsm** by double-clicking on it. You will need to allow macros to run. The on-screen instructions will guide you through the following steps

- Copy **Dexel Form.xlam** to your Addins folder.
- Excel only checks for new Addins when it starts so you will need shut down Excel and then re-open **InstallDexelForm.xlsm**
- Activate **Dexel Form.xlam**

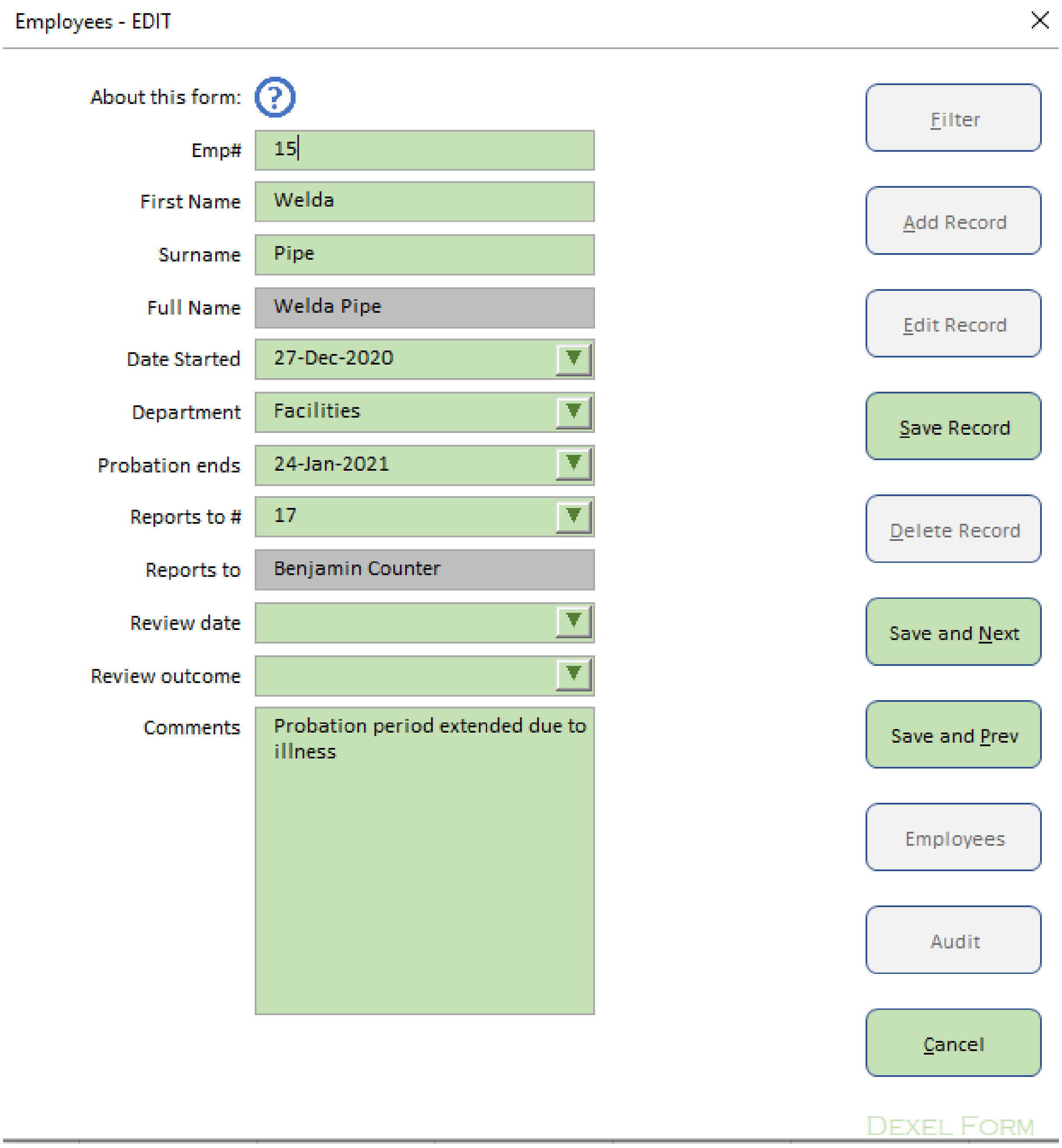
Using Dexel Form

Once you have downloaded and installed Dexel Form, the following new ribbon will appear in Excel.



You can then use Dexel Form directly on any tables you have set up in your spreadsheets.

For example when you click on **Edit Record** on the Dexel ribbon, you will get a form such as below. You can edit the data. If your edits do not comply with the defined validation rules, then offending data will be highlighted in red and you will not be able to **Save** the record until all data is valid.



Employees - EDIT

About this form:	?
Emp#	15
First Name	Welda
Surname	Pipe
Full Name	Welda Pipe
Date Started	27-Dec-2020
Department	Facilities
Probation ends	24-Jan-2021
Reports to #	17
Reports to	Benjamin Counter
Review date	
Review outcome	
Comments	Probation period extended due to illness

Filtering records

Filtering is an extremely powerful way of limiting your view of large datasets. In its simplest form, you just need to type in what you are looking for, click on **Apply Filter** and Dexel Form will do the rest. However it is possible to use Filtering in a much more sophisticated way, and the options are listed below:

When you click on **Filter**, the data entry form is displayed in filter mode. You input the selection criteria that you want to apply and click **Apply Filter**.

You can use a variety of operators and wildcards options when inputting selection criteria.

The options are :

Value	Find records where field equals Value
>Value	Find records where field is greater than Value
>=Value	Find records where field is greater or equal to Value
<Value	Find records where field is less than Value
<=Value	Find records where field is less than or equal to Value
<>Value	Find records where field is not equal to Value
<>	Find records where field is not blank
=	Find records where field is blank
Value	Find records where field contains Value
Value*	Find records where field starts with Value
*Value	Find records where field is ends with Value
Value1:Value2	Find records where field is between Value1 and Value2 (inclusive)
Value1;Value2;…	Find records where one of Value1, Value2, … is true. Each of these may be one of the operators described above.

If you input selection criteria in more than one field, they will be combined using Boolean AND.

To clear filters, click on **Filter** and then on **Apply Filter** without specifying any criteria.

Excel and Filters: Excel filters worksheet rows rather than table rows. If you have multiple tables on one sheet especially if they are beside each other, applying filters can lead to somewhat confusing results. Note also that when adding or deleting records Dexel Form sometimes needs to clear filters. Note also that verifying records automatically clears filters.

Designing your Database

To reap the full benefit of the features that Dexel Form offers, you may want to do some design work in terms of defining the data types and validations that you want the system to perform. This can be done very easily using the built-in Dexel Form features.

Dexel Form and Excel tables.

Dexel Form only works on data formatted as an Excel table. Why is this?

1. Tables do some of the work in terms of enforcing structure. For example, tables ensure that every record has the same columns (you cannot merge cells within a table). And with tables, every row has the same formulae attached to it.
2. Tables ensure that every column has a unique name. These column names are used by Dexel Form as the field names when displaying records in a form. Dexel Form also stores information that it needs (column definitions and validations) in notes associated with the column headers.
3. With tables, you can define formulae using structured referencing which is considerably more user friendly than using the standard A1 or R1C1 type of formula definition.

The main phases in designing your database are:

- Defining column properties
- Defining table properties
- Enabling protection

We go into each of these phases in the following sections.

Column Properties

The column properties that you can define for each column in the table are of the following types:

- the Basic Properties such as formatting
- the Reference Properties which are used to define the relationship with other tables in terms of cross checking of values and looking up data
- validation formulae for customising data validations
- default formulae for providing default values for new records and for duplicating records

Basic Column Properties

Property Name	**Description**	**Example**
AutoIncr	This is equivalent to an AutoIncrement option in a database. It will assign a unique readonly numeric sequential identifier to each row. A Table can have only one AutoIncr column. If you wish to enable auditing for the table, then an AutoIncr column must be defined.	
Required	Used to indicate that the user may not leave the column blank.	
Unique	The value will be unique in this column of the table. If you choose the "Warning" option, then the user will be allowed to enter a duplicate value but will be warned.	In a table of Employee records, you can specify that the Employee # is to be unique.
Composite Key	If the Unique key is Composite (i.e. consists of more than one field), then specify here which extra columns make up the unique composite key.	In a table of Employee records, you can specify that the combination of First Name and Last Name is to be unique.
Hidden	Determines if the field will be shown in the data entry form. If you leave the value blank, then the field will be hidden if the table column is hidden. If you choose Yes, the field will be shown even if it is hidden in the table. Choose No and it will not be shown in the form even if it is visible in the table.	
Readonly	The user will not be able to modify the value of the field. Fields which are the results of calculation formulae are always Readonly.	If your table has been downloaded from a database, and you want users to update only certain columns, you can restrict them from updating fixed columns by marking them as Readonly.

WriteOnce	The column can only be updated when Adding a new record. It can not be subsequently Edited.	In a table of Employee records, you may want to ensure that an Employee # is not changed after it has been assigned to a record.
Format	Dexel Form provides a number of predefined common data formats. These cover standard numeric, date, time and interval formats. Choosing a Format defines both the data type and the way in which it will be presented. So if you choose one of the Date formats, Dexel Form will validate the input as a date and will also format the input which the chosen format string. For text fields you can specify whether it is to be saved as **UPPER** or **lower** case. If you choose **Password** as the format, then the data entry form will mask user input with the * character, and the user input will be stored as plain text in the spreadsheet. If you choose **Hash** as format then user input will be masked with the * character and will be stored in obfuscated format in the spreadsheet.	
CustomFormat	You can specify a custom numeric format if one of the pre-defined formats does not meet your requirements	
Multiline	Specify the number of lines - up to 10 - that are to be used for input on the Dexel Form data entry form. This does not in itself limit the number of characters that may be input (you can use **Max** for that purpose). The user can add Return/Newline characters within the text.	Particularly useful for multiline postal address fields, comments fields, and just about any lengthy text field.
Min	The minimum number of characters that the user is required to input.	
Max	The maximum number of characters that the user is allowed to input.	
Help Message	Information message which will be available to the user when completing the data entry form.	

FormBreak	Used to indicate that this field should appear at the top of a next column on the Dexel data entry form. A maximum of 2 FormBreaks are allowed in a sheet, meaning a maximum of 3 columns on the Dexel data entry form.	If you have a large number of columns in your table, you can improve the appearance of the form by splitting it into multiple (up to 3) columns. If any particular column still does not fit on the form the Dexel Form will scroll as required.

Reference Properties

The reference properties allow you to define how the column links to other tables or named ranges. The tables or named ranges can be in a different workbook. If the table or named range is in another workbook then that other workbook must be open while Dexel Form is running or an error message will be displayed.

Property Name	**Description**
Reference Table	This is a drop down list of the tables and named ranges that Dexel Form has found in your spreadsheets. Use in combination with Reference Column to specify that acceptable values for this column are to be matched against the values stored in the Reference Column of Reference Table.
Reference Column	Used in combination with Reference Table to specify that acceptable values for this column are to be matched against the values stored in the Reference Column of Reference Table.
Display Columns	These are the extra column values that you wish to display for information purposes in the drop down list.
MultiSelect	Allow multiple selections from List. These will be stored in the table as a list separated by semicolons.
List Style	By default, the list will be presented to the user as a Drop Down list. If the number of items in the list is small, you can opt for Check Boxes. If the number of items is very large - [more than 100], then choose the No List option. With No List, the user must key in the value which will then be checked against the target table. Check Boxes may not be used in combination with a filtered list. Multi Select cannot be used in combination with No List. No List will automatically be chosen if the Reference Table contains more than 1,000 rows.
Sort Columns	If you want the drop down list to be sorted, specify here the column names by which sorting is to be done. Only ascending sorting is supported, so if you want descending sorting you will need to use a helper column or else ensure that the data is already sorted in descending order. If you leave this field blank, then the items in the drop down list will appear in the same order as in the Reference Table. The Sort Columns do not need to be in the Display Columns.
Filter Reference	Used in combination with Filter Range to create a dependent list. The list will only contain the items where the value of the Filter Reference column matches the values of the Filter Range columns in the Reference Table.
Filter Range	Used in combination with Filter Reference to create a dependent list. The list will only contain the items where the value of the Filter Reference column matches the values of the Filter Range columns in the Reference Table. The number of columns selected in Filter Range must match the number of columns in Filter Reference.
Foreign Key	If this column is specified as a List, then making it a Foreign Key will help ensure that the record to which the List refers is not modified or deleted.

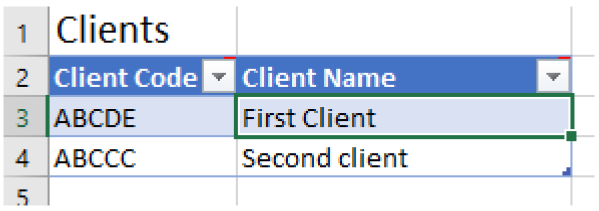
Lookup Properties

The lookup properties allow you to lookup and retrieve data from other tables using the references you have defined.

Property Name	Description
Lookup Reference	Based on one of the References that you have specified in the table, lookup the value in that column and return the value found in the Lookup Range.
Lookup Range	Based on one of the References that you have specified in the table, lookup the value in that column and return the value found in the Lookup Range.

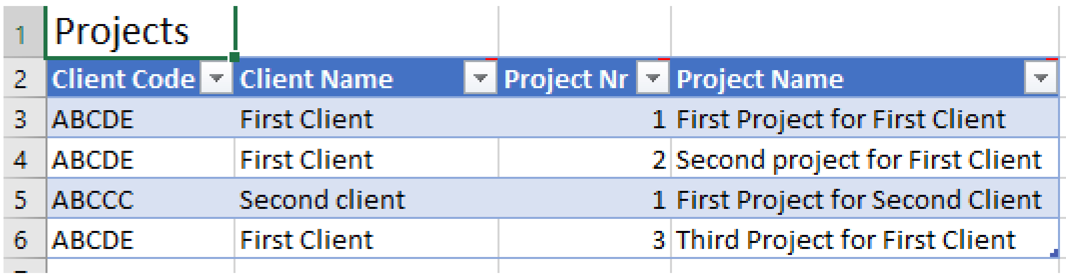
Example of Reference and Properties

In this example we have a Clients table and a Projects table.



Clients

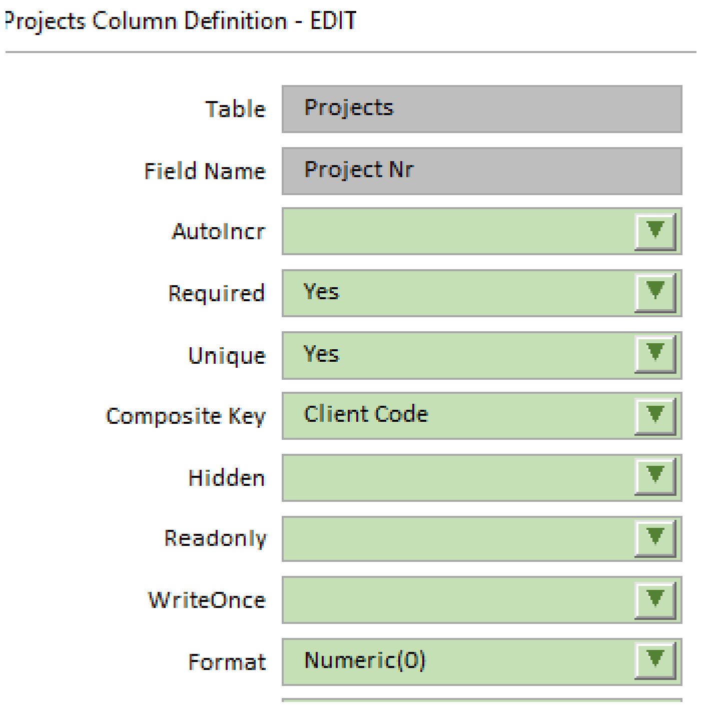
Client Code	Client Name
ABCDE	First Client
ABCCC	Second client



Projects

Client Code	Client Name	Project Nr	Project Name
ABCDE	First Client	1	First Project for First Client
ABCDE	First Client	2	Second project for First Client
ABCCC	Second client	1	First Project for Second Client
ABCDE	First Client	3	Third Project for First Client

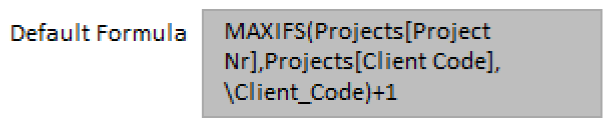
For each client we can have one or more projects. For each client the projects start numbering at 1. In the Projects table the combination of Client Code and Project Nr is unique. To ensure that, we define the Project Nr column as follows:



Projects Column Definition - EDIT

Table	Projects
Field Name	Project Nr
AutoIncr	
Required	Yes
Unique	Yes
Composite Key	Client Code
Hidden	
Readonly	
WriteOnce	
Format	Numeric(0)

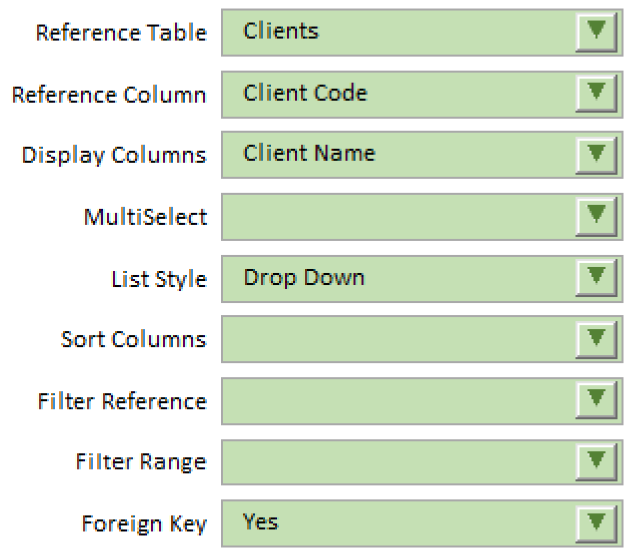
The Default formula gives us the next available Project Nr when creating a new project:



Default Formula	MAXIFS(Projects[Project Nr],Projects[Client Code],\Client_Code)+1

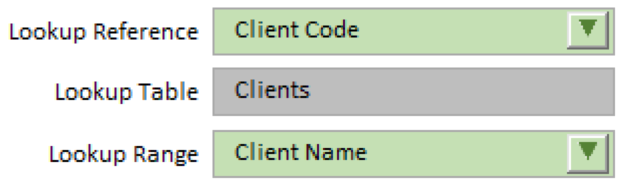
We want to link the Client Code in the Projects table to the Client Code in the Client table.

We define the Client Code column in the **Projects** Table as follows:



Reference Table	Clients	Check this table
Reference Column	Client Code	Against this column
Display Columns	Client Name	Show this extra column in the drop down list
MultiSelect		
List Style	Drop Down	
Sort Columns		
Filter Reference		
Filter Range		
Foreign Key	Yes	Do not allow the Clients record to be deleted if there is a Projects record pointing at it.

We also want to look up the client name and have it displayed in the Projects table. This is equivalent to a VLOOKUP type of formula. We define the Client Name column in the Projects table:

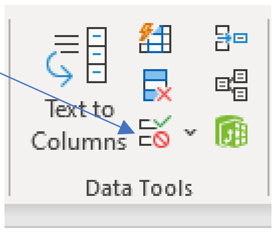


Lookup Reference	Client Code
Lookup Table	Clients
Lookup Range	Client Name

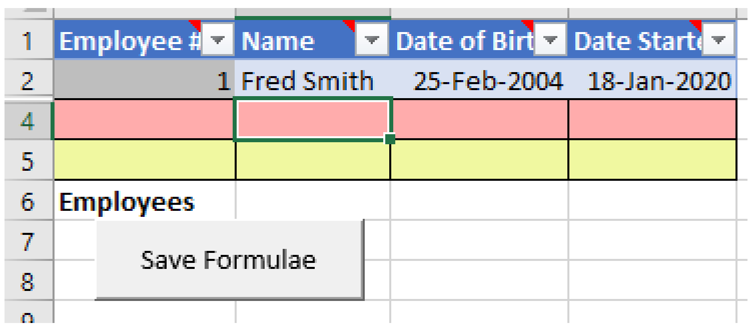
Data Validation

The following mechanisms are available for validating user input. It is possible to use all of them on the same column:

1. The Column Properties described above provide basic data type validation and cross reference validations
2. You can use the built in data validations mechanisms on the Data ribbon in Excel
3. You can define validation formulae using the **Define Formulae** option on the Dexel Form Ribbon. The remainder of this section discusses how this works.



When you click on **Define Formulae** the following screen will appear. This will contain the same columns as in the table you have currently selected in your active workbook.



1	Employee #	Name	Date of Birt	Date Starte
2	1	Fred Smith	25-Feb-2004	18-Jan-2020
4				
5				
6	Employees			
7				
8				

The validation formula for each column is input in the red row (row 4).

A validation formula is an **=IF()** formula which returns either a blank string (meaning the data is valid) or an error message which will be displayed to the user.

In the validation formula you can use structured addressing (i.e. table names and column names). A reference could take the form

TableName[@[Column Name]]

so that to refer to the **Date of Birth** in the above table you could use the reference

Employees[@[Date of Birth]]

Suppose for example we wished to validate the **Date of Birth** column to ensure that employees are at least 16 years of age. To do that the validation formula entered in the red row under the **Date of Birth** column could be:

=IF(YEARFRAC(Employees[@[Date of Birth]],TODAY())<16,"Employees must be at least 16 years of age","")

There is a shorthand version available which makes writing validation formulae a good deal simpler. Using this shorthand version, simply refer to

\Column_Name

Using this shorthand notation, the validation formula shortens to:

=IF(YEARFRAC(\Date_of_Birth,TODAY())<16,"Employees must be at least 16 years of age","")

Note that the full range of autocomplete features normally available when entering formulae in Excel are also available for Validation Formulae. In particular you can insert a shorthand version of a column name into a formula, simply click on the sample data row (the grey row (row 2)).

When you have finished entering the validation formulae, click on **Save Formulae** to return to the active table.

Default values

An important time saving device when performing Data Entry is to have the system automatically provide default values for some or all of the fields in your data entry form as you add new records. Much data entry is repetitive and often successive records differ little from each other. It is therefore necessary to have a mechanism to allow data to be either duplicated from the previously entered record, or else incremented from the previously entered record.

To accommodate this need, Dexel Form allows you to define **Default Formulae**, which can be either fixed values or values based on an Excel formula.

The mechanism for defining **Default Formulae** is almost identical to the mechanism for defining **Validation Formulae** described above. Click on the **Define Formulae** option on the Dexel Form Ribbon. The **Default Formulae** are input in the yellow row.

The shorthand mechanism described above for referring to values in the current row is available, as well as the keyword

\DuplicateLast

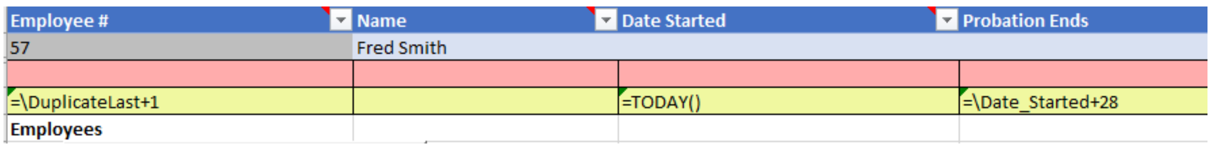
which will replicate the value entered in the last record of the table. It can also be used in expressions such as

=\DuplicateLast+1

to increment the value present in the current column of the last record in the table.

In the following example, three columns have been set up with Default Formulae:

- **Employee #** will default to one more than the value in the last record.
- **Date Started** will default to today's date
- **Probation Ends** will default to 28 days after **Date Started**



Employee #	Name	Date Started	Probation Ends
57	Fred Smith		
=\DuplicateLast+1		=TODAY()	=\Date_Started+28
Employees			

When you have finished entering the **Default Formulae**, click on **Save Formulae** to return to the active table.

Another useful built-in function is **\\UserName** which provides the user name of the currently logged in user.

Table properties

Using Table Properties you can customise the actions that users can perform on the table. For example, suppose that you do not want users to be able to **Delete** records. You can do that with **Table Properties.** To access Table Properties, click on **Table Properties** on the **Dexel Form** ribbon, to get to the following screen.

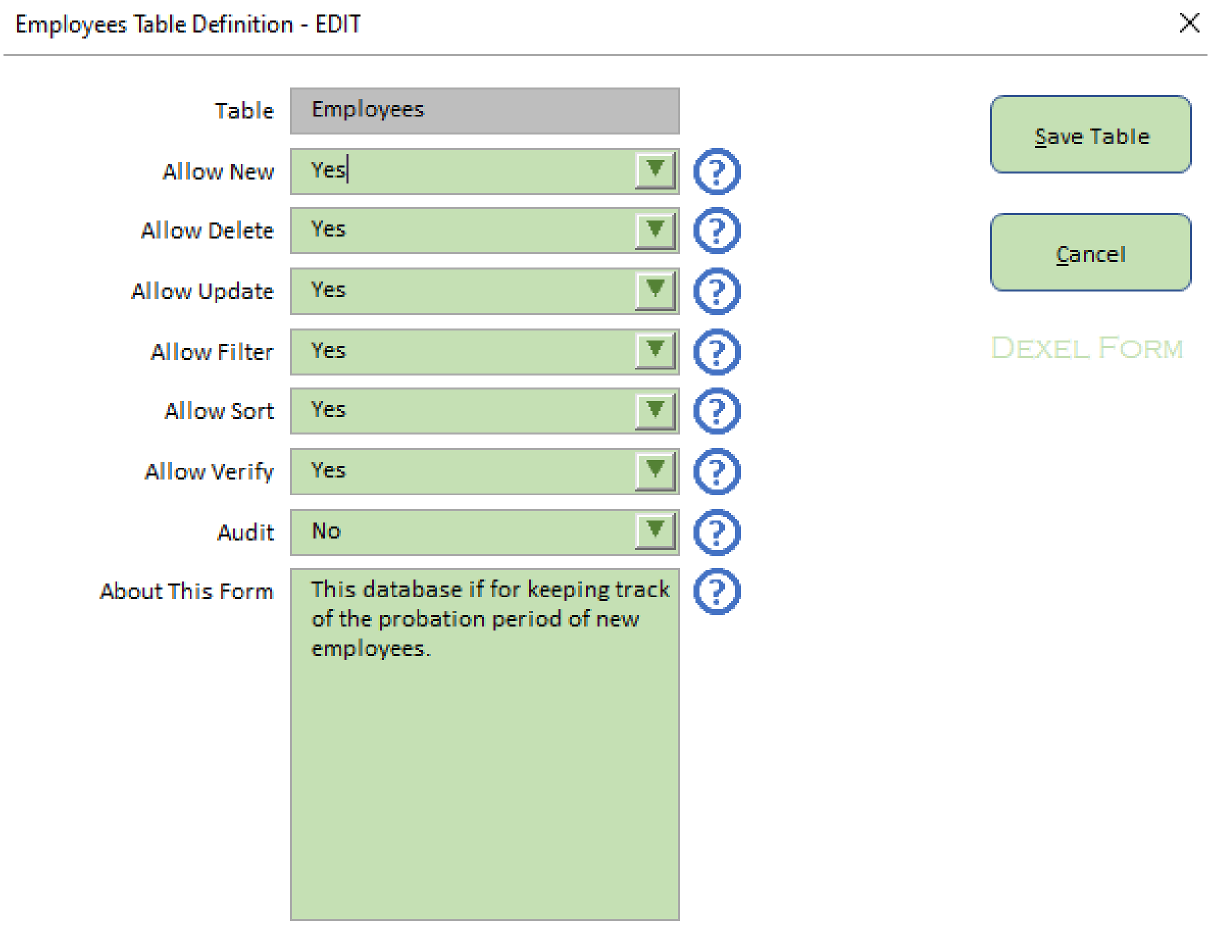


Table	Employees
Allow New	Yes
Allow Delete	Yes
Allow Update	Yes
Allow Filter	Yes
Allow Sort	Yes
Allow Verify	Yes
Audit	No
About This Form	This database if for keeping track of the probation period of new employees.

Limiting user actions

Using the drop down list beside each of the **Allow** options, you can restrict the options that the user has available in Dexel Form.

Auditing

If you enable **Auditing** for a table (e.g. the Employees table), Dexel Form will create a worksheet containing an Audit table called **\Dexel_Audit**. Dexel Form will record all changes made by Dexel Form to the Employees table. It will also add an **Audit** button the data entry form for the table so that users will have easy access to the change history of a record. If you wish to enable auditing but not allow users to view the change history, simply **Hide** the worksheet containing the \Dexel_Audit table before protecting the workbook.

About This Form

If you provide an **About This Form** text, then that will be available to users as a help button on the data entry form.

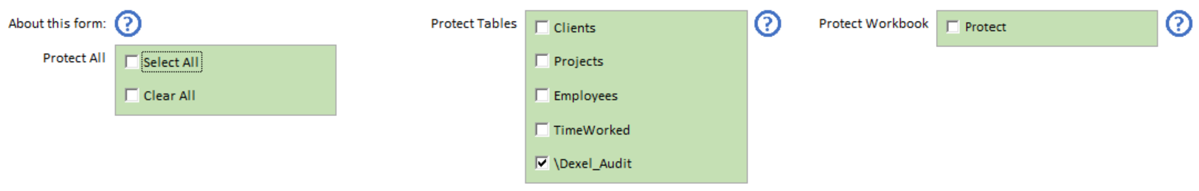
Protection

When you have finished designing your system (creating tables, defining column properties and table properties) you can ensure that the features you have chosen are rigidly enforced by enabling **Protection.**

Protection uses the built in worksheet and workbook protection mechanisms of Excel, but does this in a very convenient way. Users will not be able to update protected tables except by using Dexel Form.

If you provide a protection password, this will be stored in obfuscated form as a document property. Dexel Form will use this to unprotect and re-protect data as required.

When you click **Protection** in the Dexel ribbon, you get the following form:



Clicking on **Protect Tables** allows you to select individual tables to be protected. Note that if a worksheet contains more than one table, and you choose to protect one then all the other tables in the worksheet will also be protected.

If you choose to **Protect Workbook**, then the user will not be able to create or delete worksheets, or even hide or unhide them.

For practical purposes, the easiest option is to **Protect All**.

Note that once you have enabled protection, you will not be able to make design changes to the tables without unprotecting.

You can apply a password either before or after enabling protection. To clear a password, leave it blank.

Multi User Workbooks - Dexel Sharing

It is all very well being able to use Dexel Form to create a database like solution for collecting data, but what if you want multiple users to be able to access the data at the same time? No problem, for with Dexel Sharing you can very easily share your workbook with other users and they will be able to input and update data at the same time. All you need is a network share that your users have access to. You save your workbook on the network share, enable Dexel Sharing on your workbook and then tell your users where to find the workbook. Dexel Sharing handles the rest.

To enable or disable sharing on a workbook click on the Sharing option in the Dexel Form ribbon.

Difference between a single user and a multi user system

Excel is a single user system. It has a couple of tricks up its sleeve (workbook sharing and co-authoring) which give you some tools to help merge together the changes which different people are making to the same workbook. However these tools can be temperamental and can even lead to corruption of the workbook.

Life is much simpler if you are the only person working on a workbook. You have complete control over the data and no other user can pull the data from under your feet. The thing about a multi user scenario is that other users can do things which interfere with what you are trying to do (and vice versa). For example suppose you are using Excel as an order entry system and you are entering an order for a new customer only to discover that another user has just deleted the new customer record, or you enter an order for a product only to discover that another user has beaten you to it and entered another order for the same product and there is no more stock available. Dexel Sharing handles these conflicts for you.

If you are using Excel for a data gathering exercise or a database type of application, Dexel Form may be a perfect solution for you, especially since it handles multiple users elegantly and with no difficulty.

How does Dexel Sharing work

When you share a workbook, Dexel Form creates a custom document property in the workbook to indicate that the workbook is shared. It also marks the workbook as hidden to reduce screen flicker in later operations. We will call this shared workbook the “master workbook”. When a user opens the master workbook, Dexel Form detects that the workbook is shared. It then automatically creates a “local copy” of the master workbook and it is on this local copy that the user works. When the user adds, edits or deletes a record (a “record operation”) using Dexel Form and clicks on Save Record, Dexel Form checks on the local copy that all validations are correct. It then checks that it can get exclusive access to the master workbook. It opens the master workbook in the background, and checks that the record operation that the user is trying to perform is valid on the master workbook (i.e. that no other user has done something which makes the record operation invalid). Assuming everything is OK, the record operation is performed on the master workbook which is then closed and control returns to the user to perform his next record operation. If for some reason a conflict occurs, then the user’s local copy of the workbook is replaced with a fresh copy of the master workbook (and the master workbook is closed) and the user is asked to perform the necessary modifications to ensure that the record operation can be performed. This approach ensures that the master workbook is only locked by an individual user for a minimum amount of time. If there is a conflict where two users try to access the master workbook at exactly the same time, Dexel Form handles this conflict automatically (one user will wait while the other takes temporary precedence).

Requirements and Limitations

1. Dexel Sharing only works if all users who are accessing the shared workbook have Dexel Form installed and active. If one user does not have Dexel Form installed and active, then he will lock the master workbook and prevent other users from updating records.
2. Dexel Sharing works only on Excel tables. All updateable data in your workbook must be contained in a table. If you have data outside a table, or if you update data manually (i.e. not using Dexel Form) then your changes will only take place on the local copy and these changes will be lost.
3. The workbook will probably contain a variety of tables, some of which are dynamic in nature (i.e. users can perform record operations on them) and some of which are static in nature (lookup tables, status codes etc which users do not update). Dexel Sharing needs a unique identifier for each record in dynamic tables so that it can reconcile the changes made by different users. For this reason it is necessary to have an autoincrement column defined in each dynamic table. The autoincrement column does not have to be visible, but it does have to be present, otherwise Dexel Sharing will not permit record operations on the table. When you share a workbook, Dexel Form will detect which tables do not have an autoincrement column defined and will alert you that those tables will not be updateable after sharing.
4. Dexel Sharing does not use Excel workbook sharing or co-authoring and is not compatible with those.
5. The workbook being shared must be located on a file share. Dexel Sharing will not work if the workbook is stored on OneDrive, Sharepoint or other cloud based service. The reason for this is that the behaviour of e.g. OneDrive is timing dependent and unpredictable. It will also not work if the master workbook is stored in a Document Management System (DMS).
6. Performance may be a factor. Every time a user saves (or tries to save) a record, Dexel Form needs to open the master workbook, check the validations, update the master workbook and close it again. If you have a large or complex spreadsheet that takes several seconds to open (or if your network is slow), then you may run into difficulties if multiple users are trying to do the updates at the same time. **Note that you can dramatically improve the time taken to open and close a workbook by saving your shared workbook in binary (.xlsb) format, and there is no real downside to doing so.** There is no limit on the number of concurrent users since each user is working on his/her own local copy of the master workbook. What is more important is the total number of record operations (adds, updates and deletions) taking place, combined with the time taken to open and close the master workbook. As general recommendation, keep your master workbook lean and mean for data collection, and do the analysis, charting etc in a different workbook.
7. Protection. It is strongly recommended that you protect your tables and workbook structure (using the Dexel protection mechanism) and that you apply a protection password. This will help ensure the integrity of your data but guaranteeing that users cannot use a backdoor to update data.
8. Dexel Sharing will not work if a password is needed to open the workbook.

DexelAutomation - Programming Dexel Form

There may be occasions on which you would like to use Dexel Form to handle user input and dialogs, but you wish to control this programmatically.

Examples would be:

- You do not want the user to see all the data in the table. You may have an application in which the user is to add transactions to a table but the table itself is in a hidden worksheet. With Automation you can program Dexel Form to access hidden worksheets. You can limit the functions that the user can perform on the data (e.g. only add records).
- You want the user only to be able to see a limited view of a table. You can program Dexel Form to filter records based on criteria that you specify and then you can block the user from re-filtering.
- You may want to have a login to your application (user name and password). You can program Dexel Form to capture and validate login details.
- You may want to use Dexel Form as a dialog to allow a user to set parameters for your application that you have (such as the start and end date for a report).

For situations such as the above, Dexel Form offers DexelAutomation. You can access DexelAutomation from your own macros. In many cases you will have those macros linked to buttons in your Excel spreadsheet.

Note that with DexelAutomation you can override the Table Properties that you may have defined. For example you can programmatically add records even if your table properties have disabled adding records.

How to use DexelAutomation

A call to DexelAutomation takes the following form:

vRowNrs = Application.Run("'Dexel Form.xlam'!DexelAutomation", sTable, iFunction, vParams)

The output **vRowNrs** needs to be defined as a Variant. It will contain a one dimensional array of the row numbers affected by the call to DexelAutomation.

The input parameters are as follows:

sTable

A optional string, containing the name of the table. To reference a table in specific workbook use the format

sTable = "WorkBookName!TableName"

If the table is in the ActiveWorkbook, you do not need to specify the WorkBookName. Note that if you are referring to a different workbook, then that workbook must already be open or an error will occur.

If sTable is not provided or is left blank then DexelAutomation will use a table in the active worksheet. This is a useful option if you want to

iFunction

An integer indicating which functions DexelForm is to perform. If omitted, it defaults to 1 (Show Records). The following are valid values for one of the following integers:

1 – Show Record. The normal Dexel Form dialog will be shown.
2 – Edit Record. Dexel Form will be opened in Edit Record mode.
3 – Add Record. Dexel Form will be opened in Add Record mode.
4 – Delete Record. Dexel Form will delete the current or named record.
5 – Sort Records. Dexel Form will be opened with the Sort dialog.
6 – Filter Records. Dexel Form will be opened with the Filter dialog.
7 – Visible Records. DexelAutomation will return a list of visible records.
8 – Find Records. DexelAutomation will return a list of row nrs matching the filter criteria. These need to be direct matches.
8 – GetRecord. Returns the contents of the indicated record.

You can include the following code in your VBA module for ease of reference

```
Enum eDexelFunction
  cShowRecord = 1
  cEditRecord = 2
  cAddRecord = 3
  cDeleteRecord = 4
  cSortRecords = 5
```

```
    cFilterRecords = 6
    cVisibleRecords = 7
    cFindRecords = 8
    cGetRecord = 9
End Enum
```

vParams

vParams is an optional 2D variant array of used to pass OPTIONS and VALUES to DexelAutomation. The 2D array will have a variable number of rows and exactly 2 columns.

OPTIONS

The following options are recognised by DexelAutomation:

AutoClose - the dialog operates only on a single record. Once the user clicks Save or Cancel the dialog will close immediately. This can be used to edit a single record.

AutoExecute - the dialog happens in the background with no user interface unless a validation error occurs. This can be used with any of the iFunction values (e.g. to perform a Sort, or to Add a record). AutoExecute always implies AutoClose.

#RowNr# - Specify the row number on which you wish to operate in the table. This option is recognised by the following functions. **#RowNr#** can have any of the following values:

> **"First"** - for which the first visible row in the table will be chosen
>
> **"Last"** - for which the last visible row in the table will be chosen
>
> **numeric value** - being the row number in the table. If this number is outside the range of the table, then Dexel Form will choose either First or Last depending on the numeric value. Note that if the row is not visible (is filtered out), then the nearest visible row will be chosen as the target.

VALUES are passed as pairs being the column name and the desired value (see examples below)

For the **GetRecord** function, the output will be a 2 dimensional array of column names and values.

DexelAutomation examples

Suppose you have a table called **Customer** with the columns

Cust Nr which is an autoincrement column
Cust Code and
Cust Name

EXAMPLE 1.
The following code will silently add a record to the Customer table

```
Redim vParams(2, 1)
vParams(0,0) = "AutoExecute"
vParams(1,0) = "Cust Code"
vParams(1,1) = “C45”
vParams(2,0) = "Cust Name"
vParams(2, 1) = "Fred Smith"
vRowNrs = Application.Run("'Dexel Form.xlam'!DexelAutomation", "Customer", cAddRecord,
vParams)
```

and vRowNr(1) will contain the row number of the record added.

EXAMPLE 2. The following code will silently sort the records in the Customer table in descending order of Cust Name

```
Redim vParams(2, 1)
vParams(0,0) = "AutoExecute"
vParams(1,0) = "Sort By 1"
vParams(1,1) = "Cust Name"
vParams(2,0) = "Sort Descending 1"
vParams(2, 1) = "Yes"
Application.Run "'Dexel Form.xlam'!DexelAutomation", "Customer", cSortRecords, vParams
```

EXAMPLE 3. The following code will allow the user to edit only the last row in the Customer table, and will automatically close the dialog when the update is complete.

```
Redim vParams(1,1)
vParams(0,0) = "AutoClose"
vParams(1,0) = "#RowNr#"
vParams(1,1) = "Last"
Application.Run "'Dexel Form.xlam'!DexelAutomation", "Customer", cEditRecord,  vParams
```

EXAMPLE 4. The following code will return a list of the RowNrs of records where the Cust Name equals “Fred Smith”

```
Redim vParams(1 to 1, 1 to 1)
vParams(1,1) = "Cust Name"
vParams(1,2) = "Fred Smith"
vRowNrs = Application.Run ("'Dexel Form.xlam'!DexelAutomation", "Customer", cFindRecords,
vParams)
```

EXAMPLE 5. Inserting the following code in your macro enabled workbook in the **ThisWorkbook** object will activate Dexel Form if the user double clicks anywhere in a table in the workbook

```
Private Sub Workbook_SheetBeforeDoubleClick(ByVal Sh As Object, ByVal Target As Range, _
        bCancel As Boolean)
  bCancel = True  ' Suppress warnings about protected sheets
  Application.Run "'Dexel Form.xlam'!DexelAutomation"
End Sub
```