

Database Design

Topics in this section include:

- It all starts with your database design...
- Moving from the business requirements to a physical design
- Best Practices when defining tables and columns
- General database design ideas

Standards

Table and Attribute Naming

Guideline	Reason
Tables are named in the singular. E.g. Customer, not Customers.	This corresponds with the way we name objects. Plural names are used for collections.
Don't put blanks in any table or attribute name.	Blanks make names difficult to refer to in code and cannot be used as the name of property procedures in a class.
Spell your names using mixed case. E.g. the EmployeeProject table or the ShipToMainPhoneNumber attribute.	Mixed case is easy to read and eliminates the need for blanks or underscores in the name.
Avoid abbreviations in names unless the name exceeds about 30 characters.	There is no good reason to shorten names in since Visual Studio automatically fills in names for you as you type them in code. We have a generator that generates much of the code for names, and you can cut and paste other names.
Spell your names only with letters from the alphabet and digits. No special characters such as #, @, or _ (underscore).	Since MakeClass generates code directly from the attribute names, the names must be valid in VB as well as in the database manager.

Try to avoid using any form of the name of the table in the name of an attribute. For example, the Name attribute in a Customer table should be "Name" not "CustomerName", "CustName", etc.

An exception is when the name may be ambiguous. For example, in an Invoice table, the invoice number should probably be named "InvoiceNumber", not just "Number".

Avoid using any SQL reserved names for tables or attributes as they will may not work well in either Access or SQL Server.

Using the same spelling in every table for an attribute with the same meaning reduces learning time of a system.

ID, Group, Create, Table, By, Type, Order, Select, Release, etc.

Key Attributes

Guideline	Reason
<p>The primary key attribute is always a 4-byte integer. Its value is almost always assigned by the DBMS. (This is called an Index attribute in SQL Server, or Autonumber in Access.)</p> <p>This is true even when there's a good natural key from the business.</p>	<p>Requiring all tables to have the same data type for their key attribute simplifies manipulating tables in code. It also makes it much easier to automatically generate object code.</p> <p>No matter how reliable a natural key may seem, it's still out of control of the system designer. Even supposedly reliable keys such as Employee Number may not be available when you first set up an employee, or the format or size of an Employee Number may change during the life of your system.</p> <p>If you're concerned about preventing duplicates of a natural key, put a unique index on it. E.g. Put a unique index on a person's Social Security Number</p>

A table's key attribute is always named as the Table Name + ID. E.g. The key in the Person table is PersonID, in the Customer tables is CustomerID and so on. In Access the Caption property of the key attribute is set to "ID" only.

or on customer phone number. This guideline holds even in the case of a junction table with only two long integer keys that point back to two tables. E.g. A Junction table that tracks employee time on projects. There are two natural foreign key attributes in such a table: EmployeeID and, ProjectID. It may be tempting to make a compound key on the table using these two attributes. However, even in this case, we still create an EmployeeProjectID AutoNumber key attribute. You can still put a unique index on the combination of EmployeeID and ProjectID.

This consistency in name makes it easy to generate code.

This takes less column width in the datasheet view of the table.

Preventing Duplicate Records

With a table having an Index as its key, you may need to add an index to force uniqueness on one or more business attributes. For example, in the Customer table, CustomerID would be the key. You might put a unique index on the Name attribute to stop duplicate customers. In the EmployeeProject junction table you might put a unique index on the combination: EmployeeID + ProjectID.

Whenever you add an index to prevent duplicates, it should be named "Unique".

Mandatory Standard Attributes

All tables must have the following common attributes in addition to the key field.

Attribute Name	Description
StatusID	An int that points to a status table. This MUST be a foreign key to a status table. Most of the time, this is a foreign key to the table MCStatus.
CreateDate	Date and time the record was created. Defaults to GetDate().
CreatedBy	The login name of the person creating the record. Defaults to "System".
LastChanged Date	Date and time the record was last changed. Defaults to GetDate ().
LastChanged By	The login name of the person that last changed the record. Defaults to "System".

Other Standard Attributes

Most tables will have the following common attributes:

Attribute Name	Description
Name	The name of the object for displaying on selection screens lists, etc. If the object has a name, it should always be spelled "Name", not CustomerName, CustName, CustNam, CName, or any other variation that includes the name of the table. This attribute often has a unique index. You should always try to include a Name attribute in a table. For example, a look-up table named ProjectType. The values for Project Type might be "Customer" and "Internal". The field containing these values should be named Name, not ProjectType, Type, or Description.

DisplayOrder	An int that defines the ranking of each record in the table. This is in Lookup tables to sort records in other than alphabetical order. As a record is added to the table, the MindCrafted Framework automatically sets this value higher than any other record in the table. This column has a unique index.
Active	This attribute is used in Status tables. It is a bit that indicates if the record in the status table represents and active (1) or inactive status (0).
ShortName	A field used in Lookup table to hold an abbreviated form of the name. The short name is used for field labels and captions where there isn't enough room for the full name.
PhoneNumber	Used for phone numbers. MakeClass sets a phone number validation on any field containing the word "phone"
PostalCode	Used for zip codes or international postal codes. MakeClass sets a US zip code validation on any field containing the word "postalcode"

Junction (Intersection) Tables

Junction tables handle many-to-many relationships.

Guideline	Reason
Junction tables are named using only the names of the two tables. e.g. a junction table that tracks employees on projects would be named EmployeeProject or ProjectEmployee. Chose whichever seems the "dominate" table to go first. You can also add a qualifier to the names of the two tables if you think there may be more than one table for the junction. E.g. CustomerItemPrice if there may be more than one junction table on Customer and Item.	Tables map better to the two classes we generate for most tables: An object class named in the singular, and a collection class named in the plural.

Foreign Keys

Guideline	Reason
<p>Wherever possible, a foreign key attribute should have the same name as the key attribute in the foreign table. For example, in an Employee record, the foreign key to the Department table would be called DepartmentID. (DepartmentID is also the name of the primary key attribute in the Department table.)</p> <p>You cannot follow this standard when in one table there is more than one foreign key pointing to a single foreign table. For example, if a customer has both a sales rep and a support rep, the Customer table will wind up with two foreign keys to the Representative table. They cannot both be named RepresentantiveID. In this case, the attributes should be prefixed with a role name. For example, SalesRepresentativeID and SupportRepresentativeID.</p> <p>Note: Whenever two or more foreign keys in one table point to another table, you should consider whether you're creating a denormalization.</p>	<p>This simplifies design.</p> <p>Access automatically puts a non-unique index on any attribute whose name ends in "ID." You probably want this anyway, but you should confirm this. Otherwise, you may wind up with way too many indexes.</p>

Guided Tour of a Select Form

The job of a **Select Form** is to let the user find an object to manage.

The user can initiate Edit, Add, and Delete from the Select Form.

The Select form can also be used to allow the user to select a record and return to another form.

With and without Search

There are two basic styles for a select form, without search and with search.

Without search

Use this style when there are fewer than several hundred records.

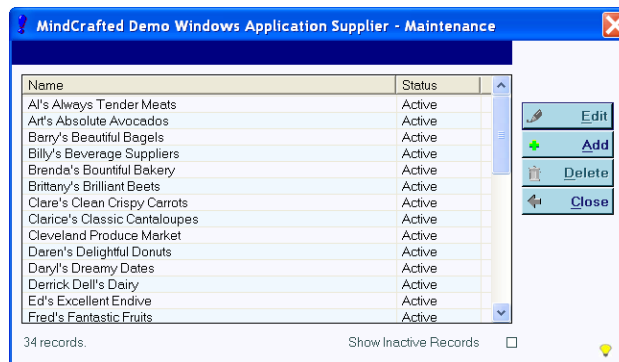


Figure 1- Select form without Search – all records are in the listview.

With search

Use this style when there are more than several hundred records.

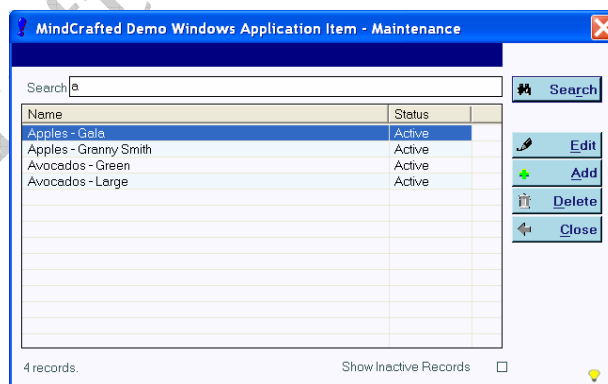
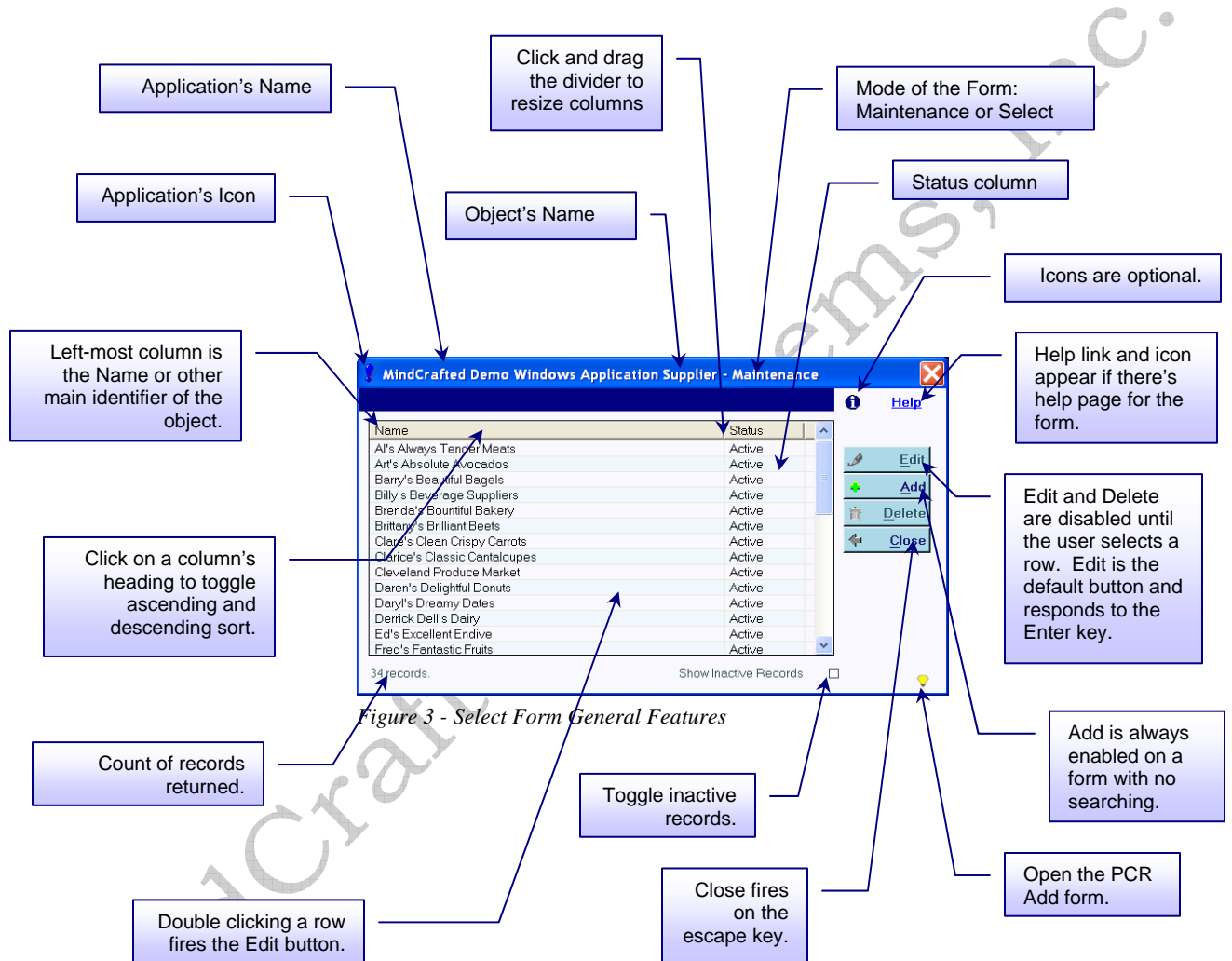


Figure 2 - Select for with Search – the user enters search criteria.

General Features of a Select Form in Maintenance Mode

Maintenance mode is the “normal” mode for a select from. It is when the form is used to add, change, and delete records.



Guided Tour of an Edit Form

The edit form is used to Edit existing records and Add new ones.

This tour uses the edit form over the Supplier table from the MindCraftedFrameWork database.

The Supplier table has a mandatory relationship from StateID to the State table. The relationship from SalesRepID to the Person table is optional. In SQL Server, the line is displayed with broken shading.

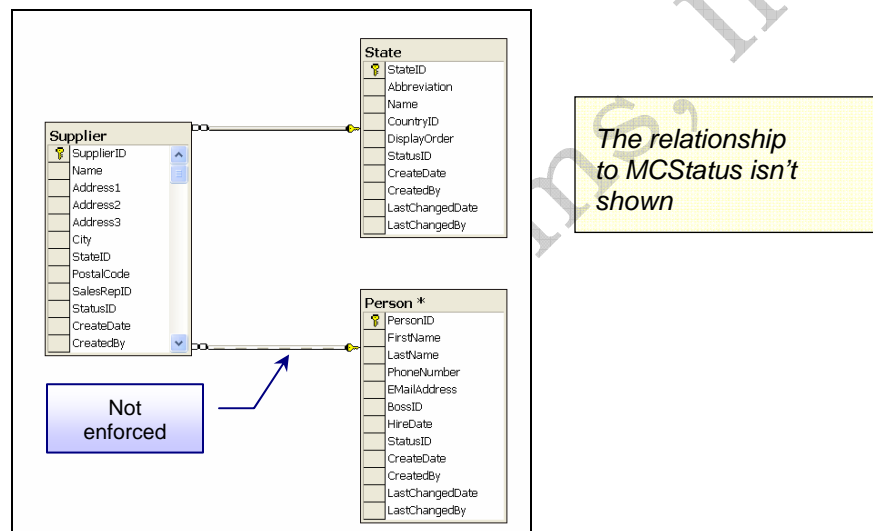


Figure 7 - Relationships behind the Supplier Edit form

General Features of a Edit Form in Edit Mode

This shows the feature of the Edit form that results from the diagram above. Not pointed out are the features that also appear on a [Select Form](#).

The screenshot shows a window titled "MindCrafted Demo Windows Application Supplier - Edit" for the record "AI's Always Tender Meats". The form has two tabs: "Basic" and "Misc". The "Basic" tab is active, showing fields for Name, Address 1, Address 2, Address 3, City, State (a dropdown menu with "arizona" selected and a "(New)" link), Postal Code, and Sales Res (a dropdown menu with "(None)" selected and a "(New)" link). On the right side of the form, there are "Save" and "Cancel" buttons. Callouts point to various features: "Required fields have a different background color" points to the Name field; "Foreign Keys get a combo box. The hyperlink on the label opens the Edit form for the record in the combo box. In this case, Arizona." points to the State dropdown; "Foreign keys that are optional have a '(None)' record inserted as the first row in the combo box." points to the Sales Res dropdown; "The Save button is the default and responds to the Enter key." points to the Save button; "The Cancel button responds to the Escape key." points to the Cancel button; "Foreign Keys have a new link the user can click on to create a new record in the foreign table." points to the "(New)" link next to the State dropdown; "Combo boxes scroll to find the record as you type." points to the State dropdown.

Figure 8 - Edit Form editing an existing record

The screenshot shows the "Misc" tab of the "Edit Form" for "AI's Always Tender Meats". The fields include: Create Date (2/11/2005 10:39:40 AM), Created By (System), Last Changed Date (3/22/2005 9:25:50 AM), Last Changed By (MINDCRAFTED01\wrank), ID (empty), and Status (Active). On the right side, there are "Save" and "Cancel" buttons. A callout points to the Status dropdown menu, stating: "The Status foreign key does not have a hyperlink or New link".

Figure 9 - Edit form's Miscellaneous tab

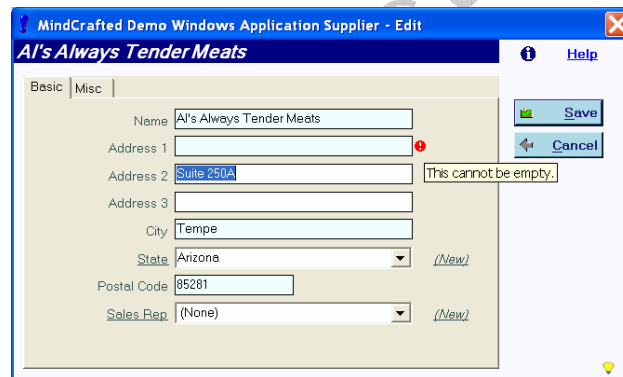
Validation on the Edit form

There are two levels of editing.

1. The form validates a field when the user tabs away from the field. If there's an error in the field, the form displays a red icon next to the field. When the user hovers the mouse over the red icon, an explanatory message is displayed. See below.
2. The data from the form is edited in the object when the user saves the record. The validations are typically the same as in the form, but the object can also perform cross-field validations.

All input fields are edited. For example, all text boxes prevent the user from entering more characters than the corresponding database field allows. Below are examples of form validations.

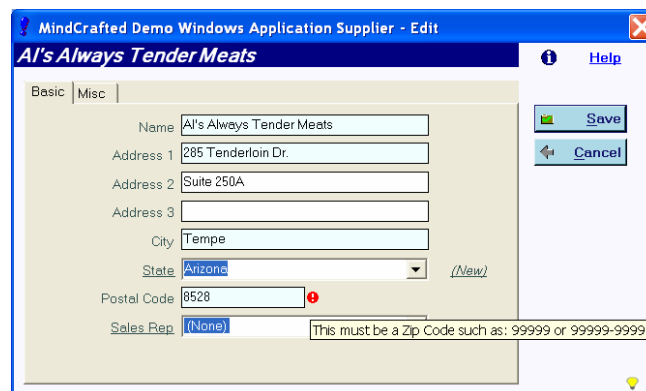
Required fields can never be blank.



The screenshot shows a Windows application window titled "MindCrafted Demo Windows Application Supplier - Edit" for the record "AI's Always Tender Meats". The form has two tabs: "Basic" and "Misc". The "Basic" tab is active. The form contains the following fields: Name (AI's Always Tender Meats), Address 1 (empty, with a red error icon and a tooltip that says "This cannot be empty."), Address 2 (Suite 250A), Address 3 (empty), City (Tempe), State (Arizona), Postal Code (85281), and Sales_Rep (None). There are "Save" and "Cancel" buttons on the right side of the form.

Figure 10 - Edit form validation for a required text box

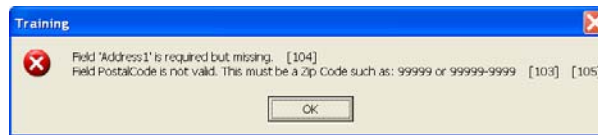
Many fields are validated for content



The screenshot shows the same "Edit" form for "AI's Always Tender Meats". The "Basic" tab is active. The form contains the following fields: Name (AI's Always Tender Meats), Address 1 (285 Tenderloin Dr.), Address 2 (Suite 250A), Address 3 (empty), City (Tempe), State (Arizona), Postal Code (8528), and Sales_Rep (None). The "Postal Code" field has a red error icon and a tooltip that says "This must be a Zip Code such as: 99999 or 99999-9999". There are "Save" and "Cancel" buttons on the right side of the form.

Figure 11 - Edit form validation for content

Clicking the Save button while there's an error on a form results in a message like this. These errors originate in the object and are displayed by the form.



MindCrafted Systems, Inc.