

3. Normalizing Data

* storing data efficiently

|--- ...When you are planning your database... ---|

Instead of thinking about what you want out ...

...think about what you have to put in and structure it well.

Okay, maybe I said that wrong, you *do* need to keep in mind what you will want to report, but that should not drive how you structure your tables; what you want out must be put in. Your table structure is driven by how things actually relate in the real world – looking at what you have from another perspective to see that can often be difficult.

Drawing an analogy to a building:

Data structure is the foundation.

Forms are walls.

Reports are windows and skylights, since they allow you to view your data.

Organize your data so the structures are strong. You want what you build on that foundation to be stable and flexible.

Structuring data is an iterative process. Set up tables, create relations between tables and print relationship diagram, change tables, print relationship diagram, put data in, change tables, print relationship diagram, ...

The best solution is simple... but it is the hardest to find.

It takes thought and foresight to design your structures well. And the simplest solution is rarely easy to see. Get the structure right before building anything else. If you have already built forms, queries, or other objects, *fix the structure before building any more!*

Just as you wouldn't build a house on sand without a strong foundation, your data structure is the foundation you will build on for Access. A good design is everything when it comes to Access. Structuring data is the most important thing to do well -- as you learn and your knowledge grows, it is important to translate that to data structure changes -- just as you would repair a crack in the foundation of a building the moment it is discovered.

It takes time to plan your database. Here are some ideas for you:

Think of the different "nouns" that you will track:

- Customers
- Addresses
- Cities
- Phones
- Notes
- Products
- Purchases
- Payments
- Campaigns

For each "noun", you describe it with "adjectives", or fields. Each table should have a primary key. It is common to use the AutoNumber field and "ID", such as CustomerID, ProductID, etc.

By convention, when "ID" is used in a fieldname, it indicates that field was originally created by an [AutoNumber](#) and it is a [Long Integer](#) data type.

Do not use "ID" as a fieldname all by itself as it is ambiguous and not descriptive.

Don't repeat data structures

If you see, for instance, that you have Department information in more than one table, create a Departments table with an [AutoNumber](#) DeptID and make a Long Integer DeptID in other tables to relate the data.

Key Fields

Key fields are used to link related tables together. The field names in the different tables should be the same (IMHO) when they represent the same piece of information. Long Integer key fields typically have "ID" in their name, such as [ProductID](#), [ProdID](#), [TransactionID](#), [TranID](#), and [CustID](#).

Data Type for Key Fields

Don't use anything but Number (field size = Long Integer) or Text for key fields – and Long Integers are most efficient. Double-precision and single-precision numbers are not accurate for keys. If you do use a text field for a key, keep in mind that it needs to be short to be efficient. It takes about 1 byte to store each character whereas long integers only require 4 bytes to store.

Starting with JET 4 (Access 2000), each text character takes 2 bytes to store in order to support Unicode. This means that a 50-character text field takes 100 bytes to store – *25 times the space needed for a Long Integer*.

DefaultValue

Change default value of all numeric foreign key fields to Null -- the default for all numbers is 0 (versions below Access 2007), which will never match with a sequential AutoNumber field -- not changing it will prevent you from being able to enforce referential integrity if it is not specified. If referential integrity is enforced, as it should be in most relationships (unless you have a good reason not to), a default value of 0 in a numeric foreign key field can cause other problems, such as append queries that fail.

To hammer that in...Access sets the DefaultValue of numeric fields to 0 -- this is not good for a foreign key because there won't be a key field to match it to : *It is okay if it is not filled out, but it is NOT okay if it is filled out with a value that doesn't match.*

For more discussion on setting the Default Value, see this thread:

<http://www.utteraccess.com/forums/showflat.php?Cat=&Board=83&Number=1411316>

Indexes

You can also create indexes on fields in tables. For instance, you may have a combination of fields that should be unique, such as TestID and QuestionID, where you would not want the same question to appear on a test more than once. You can create a unique index on the combination.

Think of which fields you will be doing lookups on and build indexes, but do this judiciously as indexes are updated when records are changed, so they take extra time to maintain, and there is a limit to how many you can create. If you use dLookup, it will work faster on an indexed field.

This topic is so important and is covered a bit more in the **Relationships** section of this document. Here is a link you may wish to read as well:

Use Microsoft Jet's ShowPlan to write more efficient queries, by Susan Sales Harkins and Mike Gunderloy

<http://articles.techrepublic.com.com/5100-22-5064388.html>

Lookup Fields in the table design

Another thing Microsoft allows you do is define comboboxes as part of the table design. Don't do this. Data should be entered via forms, which is where you would set up comboboxes.

The Evils of Lookup Fields in Tables

<http://www.mvps.org/access/lookupfields.htm>

Explore this website, mvps.org – it is an outstanding source of information.

Names

DON'T use anything but letters, numbers, and underscores in fieldnames and tablenamees.

Don't use special characters in names (% , & , / , etc). Personally, I don't even use spaces. Start all names with a letter not a number. Using numbers usually indicates that the data is not normalized anyway.

If you start a fieldname with a number, you WILL have problems, so don't ever do it. Access allows you to do many things that kick you in the butt later.

Think about how long text fields will be and set the Field size to something other than the default of 50 characters. For instance, for Americans, 30 is usually long enough for cities, 10 long enough for zips, 14 for phone numbers, 20 or 30 for last or first name. These are just guidelines, your data may need more space.

Keep names concise yet descriptive.

Name your fields and tables well. When you do create forms and reports, name the controls to something logical before building any code. Names like Text34 and Combo68 make it frustrating to code and even more so if you need help and others have to decipher what those names really mean. Personally, I like to make the **Name** property of the control the same as the **ControlSource** property whenever possible.

When you have command buttons, name them cmdClose, cmdOpenReportMenu, etc.

"Date" is a bad name for a field since that is a reserved word. Qualify fieldnames, like CourseDate and SaleDate. One reason to keep fieldnames short is consideration of the length of the field that the column heading will be over since column headings can't wrap when you open a table. Another is less to type when you are coding and ... last but certainly not least – you can fit more on a relationship diagram (covered in the **Relationships** section of this document).

more considerations:

Although convention tells us to prefix tables with "tbl", I don't like to do that...takes a split second longer to scan for values. If you do want to group your tables, use something short, like t_

Which is easier to read?

tblCustomers
t_Customers
Customers

If you are going to use a long table prefix such as "tblABC" ...make it quicker to read the important part -- the actual table name.

Which can you read faster ...

tblABCLocation
or
tblABC_Location
or
tabc_Location

always keep Names concise yet descriptive

Table and query names need to be different. To make sure they are, I do recommend using 'q' at the beginning of a query – this also groups them together when presented with a list of tables and queries as well as makes it easy to see if the source of an object is a table or a query.

Reserved Words

In addition to naming your fields (tables, queries, etc) well, you also need to avoid using special words that Access might interpret to mean something different. A list of *reserved* words, and a free utility to scan your tables, can be found on Allen Browne's website:

Problem names and reserved words in Access, by Allen Browne

<http://www.allenbrowne.com/AppIssueBadWord.html>

Allen has a lot of great information on his site, I encourage you to explore it.

Numbers that aren't numbers

If a "number" you plan to store can not be used in a mathematical expression, you may choose to store it as text, especially if it has symbols. For example, you would not do math with a telephone number, so you may wish to store it as text so that you can store the symbols and read the value easier (123) 555-1212, especially when you export your data. It is more efficient, however, to store numbers as numbers than text because the number of bytes of storage is less. Personally, I give that up for keeping the symbols.

In the case of a phone number, you could not even store it as a long integer because it has too many digits. You, therefore, would need to store it as a double-precision number and, because double-precision numbers are stored in floating point format, they are not accurate for exact comparisons.

On the other hand, a ranking, mass, or level, etc. can be used in math equations (even if you don't plan to use it that way!), so store that kind of data with a numeric data type.

InputMask

When you use the InputMask property on a text field, choose to store symbols in the field so when you do an export, they will be there. For instance, without storing symbols, a phone number would be 1234567890 when exported as opposed to 123-456-7890 or (123) 456-7890.

There are those that will disagree with me on this because it DOES make searching for phone numbers more difficult. There is a trade-off here ... in my databases, I now use a CountryCode

(from the standard list) that links to a Countries table which also stores Phone InputMask; that information is used when phone numbers are created or edited. To search on phone number, I use a function to translate the phone number to just digits without mask symbols.

Allow Zero Length

Make sure Allow Zero Length is set to Yes for text fields if you are planning on importing or using APPEND queries to add data and fields may be empty strings. You need to be careful here as a ZLS (zero length string), or even a field with a space or two, *looks* like a Null value even though it is not.

There are (many!) others who do not agree with this and consider allowing zero length to be a problem. You need to decide for yourself what you will choose to do. Here is a link which cautions you NOT to AllowZeroLength:

Problem Properties, by Allen Browne

<http://allenbrowne.com/bug-09.html>

Captions

Don't use captions for your fieldnames in your table designs! This is a bad idea. Users should never enter information directly into a table. Therefore, the only people that should be opening the table directly are administrators and using captions hides the real field name. If you want to call "ItemID" something else like TaskID, then NAME it that – and use the [Description](#) property.

There are those who do like using captions as this will also be the Caption property for a label on a form that prompts for the field information.

Descriptions

Fill out your field Descriptions! This is what the status bar text will be set to when you slide that field onto a form. The [StatusBarText](#) shows up in the lower left corner of the screen on the Status Bar when you are in that field.

Order of Fields

The Order you list fields in the table design does not matter. Usually, all key fields are at the top of the structure and tracking fields, like [DateAdded](#) and [DateEdited](#) below are at the bottom.

Tracking date record was created or modified

It is a good idea to add these 2 fields to all your tables (except tables that are used to provide values for lists and won't be altered, such as a table of states in the United States). Let them be the last 2 fields.

```
DateAdded, date, DefaultValue = Now()  
DateEdited, date -- set on the form BeforeUpdate event
```

the best way to use the [DateAdded](#) field is to set a [DefaultValue](#) of
=Now()

in the table design.

For [DateEdited](#), make sure it is your form [RecordSet](#). Then, use the Form [BeforeUpdate](#) event to set the value

```
Me.DateEdited = Now()
```

Combobox Example

Do not store names in related tables, store the IDs. For instance, instead of *storing* "Sue Smith", store the PID (PersonID) that identifies her. This can be totally transparent to the user.

In most circumstances you should not store names in more than one place. For instance, if you have a People table, define a PID (or PeopleID) AutoNumber field. Then, in other tables, when you want to identify a person, you can use the key field. One way to do this...

Create an [AutoNumber](#) field in the People table -->

```
PID, AutoNumber
```

then, in the other tables...

```
PID, Long Integer, DefaultValue: leave blank
```

Then, when you want to put data in (which should be done from a form), you can set it up to pick names from a list but store the PID. Create a combobox control:

```
Name --> PID
```

```
ControlSource --> PID
```

```
RowSource -->
```

```
SELECT
    PID,
    LastName & ", " & Firstname AS Fullname,
    BirthDate
FROM People
ORDER BY LastName, Firstname
```

```
BoundColumn --> 1
```

```
ColumnCount --> 3
```

```
ColumnWidths --> 0;2;1
```

(etc for however many columns you have -- the ID column will be hidden)

```
ListWidth --> 3.2
```

(for a combobox, this is the sum of the column widths plus 0.2 for a scrollbar)

If you have a listbox, sometimes you need to make the [WIDTH](#) .01 *more* than the sum of the columns to prevent the horizontal scrollbar from appearing.

PID will be stored in the form [RecordSource](#) while showing you names from another table... a *much* better and more reliable method.

If you want to show other information from your combobox in other controls, you can use calculated fields. For instance

```
textbox:
```

```
Name --> BirthDate
```

```
ControlSource --> = PID.column(2)
```

The reason that column 2 is referenced instead of column 3 is that column indexes start with 0, not 1, in Access.

Combobox Properties to Set

Rather than using a wizard to make your comboboxes, it is MUCH better to set the properties yourself, there aren't many of them, and then you know what was done 😊

DATA TAB

ControlSource

- Unbound if the control will be used to Find or Filter (etc) and its value will not be stored
- The name of a Field in the RecordSource of your form/report to display and edit data in your tables
- A calculation; for instance $\rightarrow =[\text{Quantity}] * [\text{UnitCost}]$
calculated controls must start with an equal sign, =

What appears IN the control when you are looking at the design view of a form or report is the ControlSource

Once you have chosen the ControlSource, copy the field name to the clipboard (use F2 to toggle between insertion point and selecting the whole value) and press Ctrl-C.

OTHER TAB

Name

This is how you will refer to your control in code, on queries, other forms, reports, etc -- so pick something logical! I like to make the Name the same as the ControlSource for bound controls.

Name cannot be the same as a fieldname in the RecordSource UNLESS it is bound to that field (in which case I recommend it)

click in the Name property and press Ctrl-V to paste the ControlSource value you copied

(back to) [DATA TAB](#)

RowSourceType

- 'Table/Query' if you are specifying a tablename (bad idea as you may change the order of your fields or modify your table structure), query name, or SQL statement as the RowSource
- 'ValueList' if you are specifying values as the RowSource
- 'FieldList' if your RowSource is a table/query/SQL and you want to list the field names

RowSource

Where Access will get the values to display for your list

I usually pick Table/Query as the RowSourceType and use an SQL statement for RowSource, something like this:

```
SELECT
    PID
    , [NameMain] & (' ' & [NameFirst]) & (' ' & [NameMid]) AS Nam
    , Code
FROM
    t_PEOPLE
ORDER BY
    [NameMain], [NameFirst], [NameMid];
```

If you are supplying a value list, such as choices for Gender --> M; Male; F; Female (assuming you will use 2 columns, and just store 1 letter)

FORMAT TAB

ColumnCount

The number of columns you wish Access to keep track of from the RowSource

The above example has 3 columns

Column indexing starts at 0, so if I want to make a textbox control to show the Code associated with a name, assuming that PID is the name of my combobox, here is the equation for ControlSource --> =PID.column(2)

ColumnHeads

True or False, depending on whether or not you want the column headings to be displayed as the first row when you drop the combo

ColumnWidths

A list, delimited by semi-colon, of the widths associated with each column

For our example --> 0";2.5";1"

The first column will be hidden

The second column uses 2.5" to display the Name

The third column will be 1" to display the code

ListRows

The number of rows displayed when you drop the combobox

By default, this value is 8

I usually change it to 24

ListWidth

How wide you want the combobox list to be

Add up the ColumnWidths and add 0.2" to allow for a scrollbar

$0 + 2.5 + 1 + 0.2 \rightarrow 3.7$ "

Width

The width of the control

In our example, the Name will show and we gave it a column width of 2.5 -- so, we should set of width of 2.5 as well (add .15" if you want to accommodate for the drop-down arrow)

EVENT TAB

OnMouseUp

While this is not a critical property to set, it is very nice to make the list of a combo automatically drop when you click anywhere in the control (as opposed to just the arrow). Here would be the code in the [Event Procedure] -->

```
me.ActiveControl.DropDown
```

OTHER TAB

StatusBarText

What will show up in the StatusBar (lower left corner of the screen) when you are clicked in the control

The display of the Status Bar can be toggled using --> Tools, Options, View tab --> 'Status Bar' checkbox

for example --> Name of Person or Company

~~~

The list above is most important for a combo but here are other properties that you may decide to set, for instance:

**Format Tab** -- Format, Top, Left, BackStyle, BackColor, BorderStyle, BorderColor, ForeColor, FontName, etc

**Data Tab** -- InputMask, LimitToList, DefaultValue, Enabled, Locked

**Event Tab** -- launch code or a macro if you want something to happen – AfterUpdate of a specific control or the form, Click, BeforeUpdate to validate a specific control or ther record before it is saved, etc.

**Other Tab** -- TabStop, ControlTipText, Tag

## Getting Help on Combobox Properties

If you do not know what some of the properties are, click in the property on the property sheet and press the F1 key for Help 🤖-- Properties are very important as they describe an object (such as hight, weight, eye color, etc describe a human)

## **More information on Normalization**

While you are designing your database, the more you read on this subject, the better ideas you will get for your own needs. As with many of the ideas here, I say one thing and others say something different so you will just need to decide for yourself what is best. Here are links you may wish to study:

### **Access Junkie's (Jeff Conrad) Resource List, Database Design 101**

<http://www.accessmvp.com/JConrad/accessjunkie/resources.html#DatabaseDesign101>

### **A Little Treatise on Normalization, by Jerry Dennison**

<http://www.utteraccess.com/forums/showflat.php?Cat=&Board=53&Number=197282>

### **Data Modeling, by George Hepworth (GroverParkGeorge)**

<http://www.gpcdata.com/downloads/normalizationgpgonaccess.pdf>

### **Glossary of Terms and Definitions, by Jerry Dennison**

<http://www.utteraccess.com/forums/showflat.php?Cat=&Board=53&Number=213518>

### **General Primary Keys - Natural vs Surrogate, by Glenn Lloyd (argeedblu)**

<http://www.utteraccess.com/forums/showflat.php?Cat=&Board=68&Number=637301>

### **Microsoft: Object Hungarian Notation Naming Conventions for VB**

<http://support.microsoft.com/kb/q173738>

### **Rules of Data Normalization**

<http://www.datamodel.org/NormalizationRules.html>

### **Training Day 1 (TD1): Introduction to Relational Database Design, by The Gunny**

[http://www.access.qbuilt.com/html/rdb\\_design\\_boot\\_camp.html](http://www.access.qbuilt.com/html/rdb_design_boot_camp.html)

### **Training Day 2 (TD2): First Normal Form, by The Gunny**

[http://www.access.qbuilt.com/html/rdb\\_design\\_boot\\_camp1.html](http://www.access.qbuilt.com/html/rdb_design_boot_camp1.html)

There are a lot of things to read! And it may be hard to justify the time on design when your boss wants a database he can use ... but as the familiar saying goes: "*Measure twice cut once*". With Access, it is more like "*Measure 50 times, cut once*". The time you spend up-front creating a solid structure will save you endless headaches down the road.