

Mistakes beginners make with Microsoft Access

Just getting started with Access? There are four common mistakes to avoid.

By Jackie Grubb (jackie@plumsuite.com)

The first thing you have to do when working with Access is to design your data tables. If they are not adequately designed to begin with, you may run into problems as the database grows and develops. Just about everything else you do in Access depends on the structure of these tables.

There are four common mistakes that database novices make when designing Access tables:

1. They combine data in one field that should be in two or more fields.
2. They assign the wrong data type to a field.
3. They use the default size for text and number fields.
4. They try to put all the information into one table.

Let's take a look at each of these common mistakes in more detail.

Combining data in one field

Having one field for city, state and zip code is an example of a common mistake that beginners make. Three fields are needed in this case, and they can be easily recombined for mailing addresses. One of the big advantages of having a database is the ability to sort and filter on a particular field. It would be difficult, for instance to put your mailing labels in zip code order in a combined field. Or, if you wanted just to mail to people in the state of New Hampshire that would also be difficult to do with the data in a combined field.

Assigning the wrong data type

Access includes the following data types:

- Text
- Number
- Currency
- Date/Time
- AutoNumber
- Yes/No
- Memo
- OleObject
- Hyperlink

A common mistake that beginners make is to assume that a zip code should be a *Number* field. New England users find out about the problem with this faster than do users in other parts of the USA. My zip code in Boston is 02215. If I enter it as a *Number*, the leading zero gets dropped and my zip code becomes 2215. The post office does not know what to do with that. A simple rule is to use a number field when calculations (add, subtract, etc) will be needed. Otherwise, make it a *Text* field.

Another common mistake is to use a *Text* data type for dates. While this works up to a point, it is better to use the *Date/Time* data type. That way you will be able to do date calculations. For instance, you might want to calculate someone's current age from their birth date, or determine how many payments are two months past due. Using the *Date/Time* data type also allows you to sort in chronological order or filter for a particular date range.

Using field size defaults

The Text data type frequently defaults to 50 characters. A default is what you get if you don't change it. In this case you would get a field that takes up to 50 characters. I have seen many databases where all the text fields are set to 50 – but that is not always an appropriate size. If you have a database with just US addresses in it, two characters are adequate for a state field. More than that is a waste of disk space. Fifty characters, however, may not be long enough though for a field that describes something in detail. You may use up to 255 characters in a text field. If you need more than that you will need to use a Memo field instead. Keep in mind, however, that Memo fields have limited sorting and filtering capability – so use them only when sorting and filtering is not important.

To give you an idea about text length: The paragraph above this one would not fit into a *Text* field. This paragraph however, would. It is 169 characters long.

When choosing the size of a *Number* field you need to consider how precise you need to be about decimal places. The default size is usually *Long Integer* – which does not allow for any decimal places. If you are dealing with money, use the *Currency* data type. Otherwise, check out Microsoft's documentation for the appropriate size to meet your particular need.

Putting all data into one table

To illustrate what we are talking about here, let's assume a situation where customers are to be invoiced. It might be tempting to put the customer's name and address information in the same table as the invoice information -- such as its date and amount due. The first problem with this is use of disk space as you would be repeating the same customer's name and address each time you create an invoice. That, however, is the least of your problems. Suppose that you have two invoices for the same customer. On one the street address is 51 Main Street. On the other it is 501 Main Street. – or a completely different address due to a move. How do you know which one is the correct address?

The problem starts getting really muddled when you start adding line item information to the invoice. Perhaps you are beginning to see that the issue we are talking about here is repeating sets of the same data. Solving that problem is one of the big advantages to working with a relational database such as Access.

The solution to the problem in the situation described above is to have three tables: customer, invoice, and line item.

The three tables are linked by unique identifiers. So, in the customer table you would have a *CustomerId* field that is unique to each customer. Access provides any easy way of creating this type of field using the *AutoNumber* data type. A different unique number is automatically assigned each time a new record is created. (Note: One to a customer!)

The invoice table would also have its own unique number in an *InvoiceId* field – one to each invoice, but it would also have a *CustomerId* field to link it to the correct customer. This would mean a one-to-many relationship – that is one customer, many invoices (assuming business is good!). Records in the line item table would be linked in a similar way to the invoice table.

That's how you break it down. database jocks look at the situation above as three generations of tables, the customer table being parent to the invoice table and the invoice table being parent to the line item table. Access provides many convenient ways of joining or linking the tables together to make them look the way we want to see them on screen and in reports.

This has been a very simplified explanation of what is called *normalization* in database jargon. I tell you this because there has been a lot written about this topic in books about databases. So that is the word you would look up if you want to know more.