DATABASE 101

What exactly is a database and how do databases work? What's the difference between a spreadsheet database and a "real" database? Read on to find out!

What is a database?

The simple answer to this question is:

> "A database is an organised collection of information".

DATABASICS

You use databases all the time, although you might not realise it! For example:

- Telephone directories
- The list of contacts on your phone
- Online product catalogues and shops

The important thing about databases is the way that the information is organised. Let's look at a simple telephone directory as an example. It has entries like this:

> Bloggs, Joe 1234 Main St. ..... (123) 123456

You could simply enter all the information as shown here and then print your directory. The problem is that if you wanted to do something else with the data - for example, organise it in a different way; find all the people in the 123 area code, or publish a different version of the directory, you would have a lot of work to do. But if the information was orgranised in a database, those tasks would be very easy.

Our example listing consists of five different pieces of information:

- Last name
- First name
- Address line
- Area code
- Phone number

We refer to each of these bits of data as a *field*.
Each group of fields belonging to the same item (product, person, company, etc.) is called a *record*.
And each collection of records of the same type of data is called a *table*.

A good analogy is a card index file, such as a Rolodex:

- The whole Rolodex is equivalent to a table
- Each card is equivalent to one record
- And each individual piece of information on each card is a field

Another (more modern!) analogy is a spreadsheet:

- One worksheet is a table
- Each row in the worksheet is a record
- Each column is a field

When you look at a list view of a table in a database such as CatBase, it looks very much like a spreadsheet.

RELATIONAL DATABASES

The structure described above is what we refer to as a "flat-file" database: it's a one-dimensional collection of data about one topic. This is fine for many purposes, but frequently we need to maintain more complex structures in which data from two or more tables is linked together. Typical examples include:

- Families (parents and children)
- Products and Parts
- Manufacturers and Products
- Classes and Students
- Customers and Invoices

You might be tempted to think "Why can't I just add more fields if I need more information?" For example, if your database contains information about products and you want to add a list of the parts that make up each product, why not just add fields for "part 1", "part 2" and so on? Well, yes, you can, but you'll run into limitations such as:

- Some products might have only one part; another might have 20. How do you know how many fields to add?
- What if the same part is used in a number of different products?
- What if you wanted to find a particular part? You would have to search all those additional parts fields, because you wouldn't know which one it might be in!
- What if you wanted to produce a report listing all your parts and their prices? That would be extremely difficult.

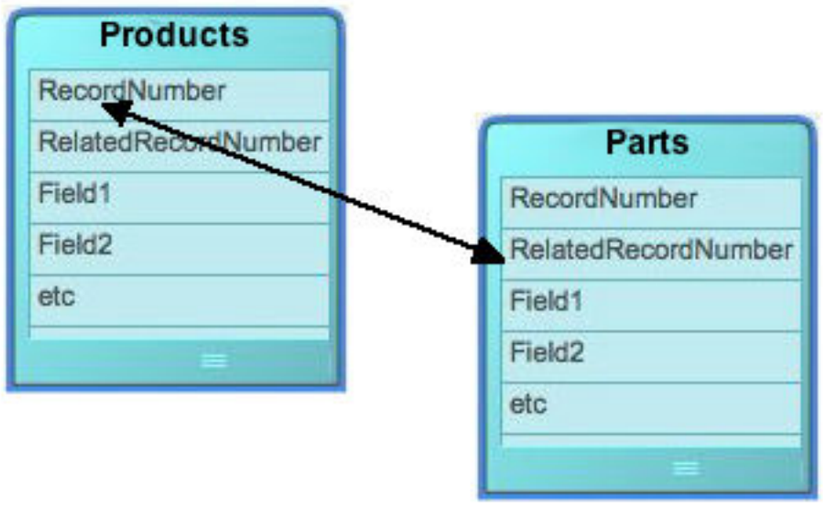
That's why we have relational databases. In our products and parts example, we would have two tables: **Products** and **Parts**. The basic product details would go into fields in the Products table, and all the details about each part would go into a record in the Parts table. We refer to the "one" table (Products in this example) as the *parent table*, and the "many" table (Parts) as the *child table*. One more important thing is needed: something to tell the database how the records are linked together - in other words, which Parts belong to which Product (or Products).

Unique Identifiers

Each record must have a **unique identifier**. It can be simply a sequential number, or an alphanumeric code of some sort. The important things are that it must be UNIQUE and it must never change. In CatBase, there are two unique identifiers for each table: the *RecordNumber* field and the *UUID* (Universal Unique Identifier) field. The RecordNumber field is an integer (whole number), and it is unique within that table. The UUID field is a string of 36 characters and it is UNIVERSALLY unique. The child records are linked, or related, to their parent records via a foreign key field. In CatBase, the foreign key field is named RelatedRecordNumber. (Note: These are the default field names. You can change the field names if you want to.)

Learn more about record numbers and UUIDs.

The following illustration shows a setup for a many-to-one relationship between our Products and Parts tables:



Many Parts can be related to **one** Product by matching the *RelatedRecordNumber* in the Parts table to a *RecordNumber* in the products table. So, to find all the parts that comprise the product whose *RecordNumber* is 1234, you would search the parts table for all records whose *RelatedRecordNumber* is 1234. There might be just one, or there might be dozens. Fortunately, with CatBase you don't need to be concerned with these technical details - CatBase manages these relationships automatically for you. There's an easy way to find all the Parts that belong to a Product:

1. In the Product table list view, select the Product (or Products) whose Parts you wan to see.
2. Right-click or Ctrl-click on the list, with the selected record(s) highlighted.
3. Choose **Show related**->**Parts** from the pop-up menu.
4. CatBase will find the Parts and display them in a list in a new window.

Many-to-Many Relationships

But what if the same part can be used in a number of different products? That's what we call a many-to-many relationship: many Parts can be related to many Products. To facilitate this, we just need to tell CatBase that we want the relationship to be many-to-many instead of the default many-to-one:

1. Go to the **Admin** menu and choose **Database Setup**->**Table and Field Setup** from the topics on the left of the page.
2. Click on the **Manage Relations** tab.
3. Double-click on the line that joins the two tables (the Products and Parts tables, in our example).
4. Select the **Many to Many** radio button.
5. Click the **Save** button.

For a more detailed explanation of how CatBase manages relations, and how to import related data into your CatBase database, see the Relations Tutorial.