MapInfo

Databases and Thematic Mapping

Oxford University
Computing Services

How to Use this Course Book

This handbook accompanies the taught sessions for the course. Each section contains a brief overview of a topic for your reference and then some contain one or more exercises.

Exercises are arranged as follows:

- A title and brief overview of the tasks to be carried out;
- A numbered set of tasks, together with a brief description of each;
- A numbered set of detailed steps that will achieve each task.

Some exercises, particularly those within the same section, assume that you have completed earlier exercises. Your teacher will direct you to the location of files that are needed for the exercises. If you have any problems with the text or the exercises, please ask the teacher or one of the demonstrators for help.

A number of conventions are used to help you to be clear about what you need to do in each step of a task.

- In general, the word **press** indicates you need to press a key on the keyboard. **Click**, **choose** or **select** refer to using the mouse and clicking on items on the screen.
- Names of keys on the keyboard, for example the Enter (or Return) key, are shown like this ENTER.
- Multiple key names linked by a + (for example, CTRL+Z) indicate that the first key should be held down while the remaining keys are pressed; all keys can then be released together.
- Words and commands typed in by the user are shown `like this`.
- Labels and titles on the screen are shown like this.
- Drop-down menu options are indicated by the name of the options separated by a vertical bar, for example File|Print. In this example you need to select the option Print from the File menu. To do this, click with the mouse button on the File menu name; move the cursor to Print; when Print is highlighted, click the mouse button again.
- A button to be clicked will look like this.
- The names of software packages are identified *like this*, and the names of files to be used like this.

Software Used

MapInfo Professional v9.5

Windows XP

Excel

Notepad

Files Used

Sites_EnglandWales_1.xls

Sites_ScotlandNIreland_1.txt

UKIreland_Admn_poly.DAT

UKIreland_Admn_poly.ID

UKIreland_Admn_poly.MAP

UKIreland_Admn_poly.tab

UKIreland_RegionalCounts.DAT

UKIreland_RegionalCounts.tab

Revision Information

Version	Date	Author	Changes made
1.0	February 2008	Michael Athanson	Created
1.1	February 2009	Michael Athanson	Updated for MapInfo v9.5

Copyright

Contents

1 Introduction

Welcome to the course **MapInfo Level 3: Databases and Thematic Mapping.**

This booklet accompanies the course delivered by Oxford University Computing Services (OUCS), IT Learning Programme. Although the exercises are clearly explained so that you can work through them yourselves, you will find that it will help if you also attend the taught session where you can get advice from the teachers, demonstrators and even each other!

If at any time you are not clear about any aspect of the course, please make sure you ask your teacher or demonstrator for some help. If you are away from the class, you can get help by email from your teacher or from help@oucs.ox.ac.uk

1.1. What You Should Already Know

This session is the second of a series that covers the fundamentals of using *MapInfo,* which is a Geographic Information System (GIS).

MapInfo Level 2: Fundamentals, or an understanding of the material it covers, is pre-requisite to this course. We will assume that you already have a basic working knowledge of *MapInfo* and its file structure and user interface. You should already be familiar with how to open/import files in *MapInfo*, and how to view and edit them using map windows and the Layer Control.

The computer network in OUCS may differ slightly from that which you are used to in your College or Department; if you are confused by the differences, ask for help from the teacher or demonstrators.

1.2. What You Will Learn

In this session we will cover the following topics:

- The relationship between map objects and database records
- Importing existing databases into *MapInfo*
- Creating databases from scratch in *MapInfo*
- Updating and querying databases using expressions
- Projecting coordinate data and creating map objects
- Associating database records with existing map objects
- Thematising map objects using colour-coding and symbology

Topics covered in related *MapInfo* sessions, should you be interested, are referenced where applicable.

1.3. What are Databases and Thematic Maps?

A database is essentially nothing more than a structured collection of data. While a database can, strictly speaking, exist in a 'hard copy' form, *e.g.* as a printed table of data, the term 'database' is normally used to refer to data in a *digital* form. A key distinction to make is between the database itself, which is the content, and the database management system (DBMS), which is the software that is used to read, edit, query, analyse, and store the database. In other words, the database is the file and the DBMS is the program or application that opens it. Perhaps the most widely recognised examples of desktop DBMS software are *Excel* and *Access*. While *Access* is a true relational DBMS, *Excel* is technically a spreadsheet application.

However, most GIS, including *MapInfo*, have some DBMS functionality. In *MapInfo*, you can not only open existing database files, but also create databases from scratch and edit, query, and update them. As a GIS user, you therefore have two options: working with your data in a dedicated DBMS first, then moving it into GIS; or creating your database directly in GIS. In this session, you will try both.

A database is a *structured* collection of data, and it is perhaps easiest to think of this structure as being like a table, which is organised in rows and columns. In such a database, each row is usually referred to as a 'record,' and each column as a 'field.' In the sorts of databases that are applicable to GIS, each record normally represents a spatial entity of some sort, be it a person, object, place, area, *etc*. The fields then represent that entity's attributes (*e.g*. identity, type, size, magnitude, date) and often – but not always – some reference to its geographic location (*e.g*. coordinates, postcode, place name, region).

A key strength of GIS is its ability to associate database records with graphical map objects. The map objects can then be 'thematised' according to any of the fields for that record – in other words, various aspects of the objects' appearance (*e.g*. colour, size, symbol) can be modified to code or represent these fields according to some underlying logic, *e.g*. a colour-coding scheme, a size graduation scheme, or a categorical symbology. A thematic map is therefore a map that uses this kind of modification to represent spatial entities' attributes.

1.4. What is MapInfo?

MapInfo is a geographic information system (GIS) that is very popular among entry-level users due to its low cost and ease of use.

GIS is software that is designed to store, query, analyse, process, and visualise geographic data. *MapInfo* can be used, for example, to plot distributions onto base maps; to create colour coded or other 'thematic' maps; to extract data from printed maps or aerial imagery; to model, analyse, and visualise elevation data; or to calculate correlation statistics between spatial datasets.

A key feature of GIS is its capability to associate spatial and attribute data. It is perhaps useful to think of it as being partly like a database management system (*e.g*. *Excel* or *Access*) and partly like an image processor (*e.g*. *Photoshop*).

2 The relationship between map objects and database records

It is useful to understand the relationship between map objects and database records from the start of this session. Note that databases are called 'tables' in *MapInfo*.

Records in a geographic database contain both spatial and attribute data – spatial data expresses *where* the record refers to, and attribute data describes *what* is located there. A key strength of GIS is its ability to relate records in a database with objects on a map. The map objects are, in other words, a visualisation of the records in the database.

Each GIS handles the relationship in a slightly different way. In *MapInfo*, the rules are a bit complex, but they can be summarised as follows:

- There is normally a 1-to-1 relationship between table records and map objects (Figure 1)
- It is possible for table records to not have any corresponding map objects (Figure 2)
- It is NOT possible to draw objects onto a table's map without creating records for them (Figure 3)
- When you draw objects onto a table's map, an empty record is automatically created for each object (Figure 4)
- If you draw map objects onto the Cosmetic Layer, no corresponding records will be created, since there is no table for that layer (Figure 5)

Each of these cases will now be described in more detail.

2.1. Table records with corresponding map objects

There is normally a 1-to-1 relationship between table records and map objects. That is to say, for each object there is one (and *only* one) corresponding record in a table, and *vice versa*. This is what GIS is designed for, and it's what distinguishes it from software that is solely for database management, or solely for graphics.

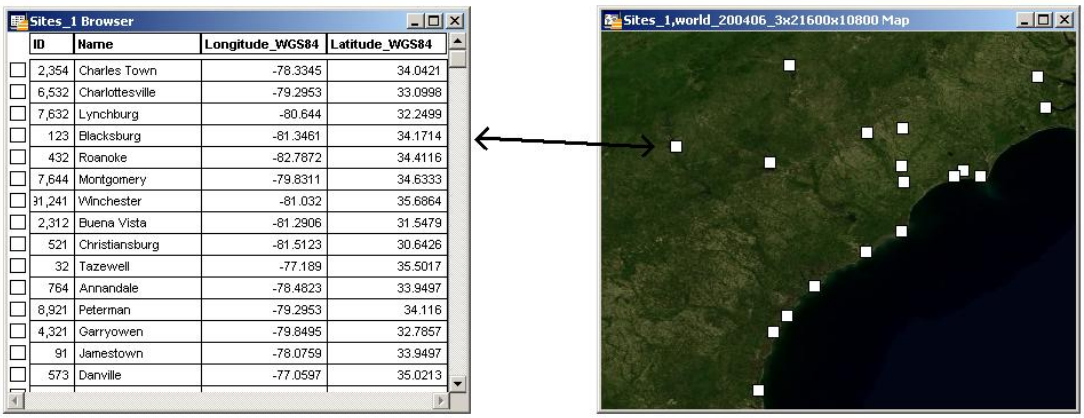

Figure 1 There is normally a 1-to-1 relationship between table records and map objects

In this situation, MapInfo enables you to explore the data in very useful ways. When you make a selection (using any of the graphical tools, or by running a

query) the selected record and its corresponding object will both be highlighted. Also, there is a graphical Info tool that allows you to click on map objects and view the contents of the corresponding record.

If you have a table and you want to know whether there are any associated map objects, go to **Window | New Map Window** and open a new map with only that layer in it. Then right-click in it an go to **View Entire Layer | All Layers**. If the message '"No Objects Found in Layer" appears, there are none. If there are any, *MapInfo* will frame them all in the map window.

If you have map objects and you want to know what table they're in (or whether there even is a table – they could be cosmetic), try clicking on them using the Info tool. The **Info Tool** window will pop-up, and in the lower-right corner of the window it will display the name of the table the map object refers to.

2.2. Table records without corresponding map objects

When you create a table from scratch within *MapInfo*, or when you import a database from a non-GIS source (e.g. an *Excel*, *Access*, or text file), there will not be any associated map objects. You will need to create these yourself, and how you do it will depend on what data you have. There are two basic approaches. If you have coordinate data, you can plot map objects simply by going to **Table | Create Points** and entering a few parameters. You will end up with one point for each record. On the other hand, if your spatial data is references to geographic features (*e.g.* administrative regions, place names, rivers, *etc*) then you will have a bit more work to do. First, you will need to find a dataset that contains these features as map objects, and second, you will need to associate the records in that table with the records in your table. This kind of integration can be difficult, but straightforward cases should not pose a problem. In this session, you will try both approaches to creating map objects.

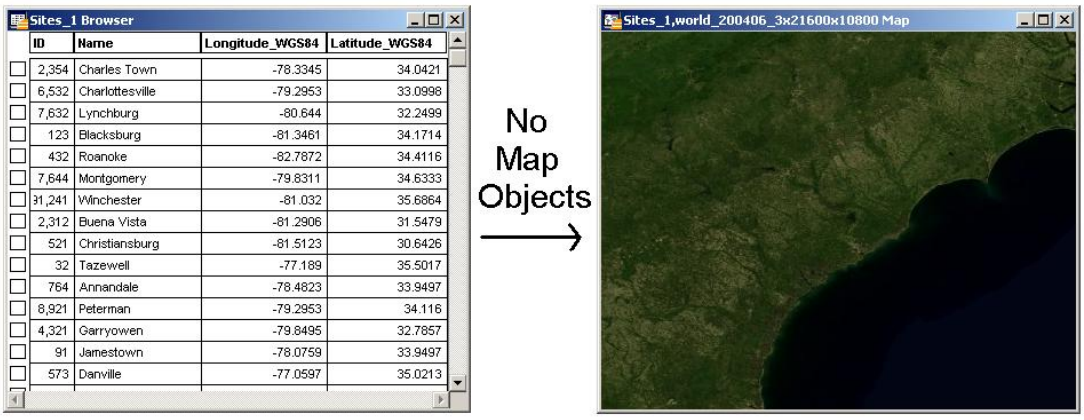

Figure 2 It is possible for table records to have no corresponding map objects, e.g. when a table is created from scratch, or imported from a database. In this case, the objects must be created (projected onto the map) from the data

2.3. Map objects without corresponding table records

It is not possible to create map objects for a table without creating records for them. When you draw objects onto a table's map, an empty record is created for each object. However, if you draw objects in the Cosmetic Layer, there will be no corresponding records for them, because there is no table for the Cosmetic Layer. You can, of course, save the contents of the Cosmetic Layer to a table, in which case a record will be created for each object.

Figure 3 It is NOT possible to create map objects for a table without creating records for them, e.g. when you draw objects onto a table's map, an empty record is created for each

Generally speaking, this kind of case will apply if your method of capturing spatial data is graphical, *i.e.* through the map. For instance, if you have a raster such as a scanned map or an aerial photograph, you can draw points, lines, polygons etc on top. This is called 'heads-up' digitisation, but think of like tracing. When you're done, you can get rid of the raster and you have the extracted vector drawing (tracing) left.

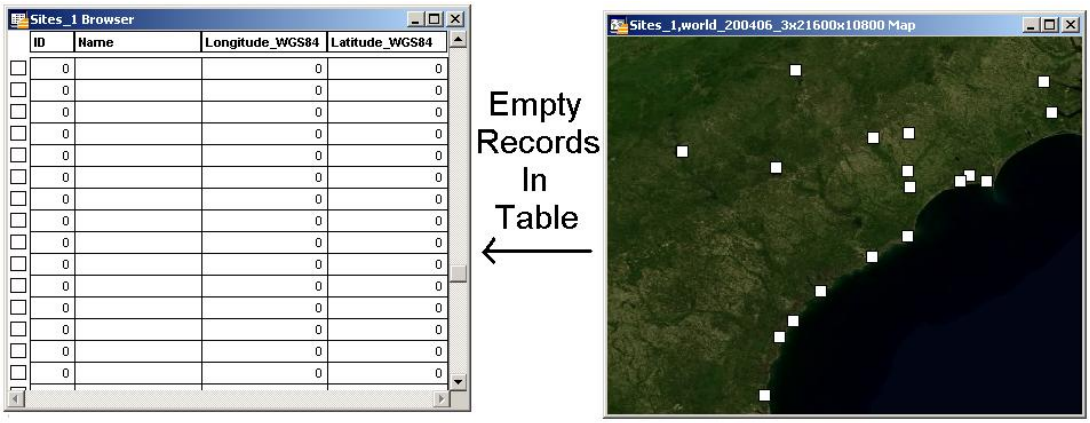

Figure 4 When you draw objects onto a table's map, an empty record is created for each object. In this case, the Latitude and Longitude columns can then be automatically populated using the Coordinate Extractor plug-in

Regardless of how you've made your drawing, if it includes point objects, you can then automatically extract coordinate data for them by running a plug-in called *Coordinate Extractor*. This process is essentially the reverse of the Create Points function – you're going from map objects to coordinate data. Heads-up digitisation will be covered in the MapInfo Level 3 session on raster imagery.

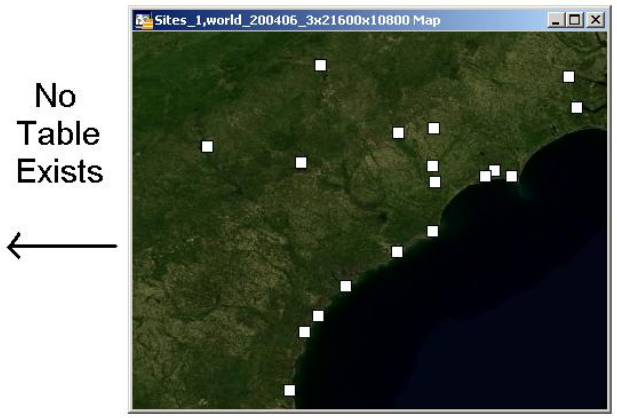

Figure 5 If you draw map objects into the Cosmetic Layer, there will be no corresponding records for them, because no table exists for that layer. Of course, you can save the contents of the Cosmetic Layer to a table.

3 Importing Existing Databases into MapInfo

If you have an existing database file, chances are good that *MapInfo* will be able to either open it directly (go to **File | Open** and pull down **Files of Type** for a list of compatible formats), import it (**Table | Import**), or translate it using one of the plug-in tools (**Tools | Tool Manager**), such as the *Universal Translator*.

On the other hand, you may have to enter your database yourself. As a GIS, *MapInfo* is partly a database management system (DBMS) and partly an image processor. However, it's not *really* ideal for either job. So, while it is possible to create your database (*i.e.* table) from within *MapInfo*, this is only really advisable if it's a relatively small amount of data. If you have a lot of data to enter, or if you need to do lots of fancy database processing or editing, it is better to do it in a dedicated spreadsheet (*e.g. Excel*) or DBMS (*e.g. Access*) first, and then bring it into *MapInfo*.

Exercise 1 Importing Existing Databases into MapInfo

- *In this exercise, you will practice opening two existing databases in MapInfo. One is an Excel spreadsheet, and the other is a delimited text file.*

Task 1 Ensure that the file permissions on all of your files are set to both 'Read' and 'Write' You will not normally need to do this when you are working on your own machine, but on the classroom network it is likely to cause problems. *MapInfo* automatically re-writes some files after certain processes, and it will generate an error if it is not able to do so	**Step 1** Browse to the **MapInfo_Databases** directory
	Step 2 Right-click on the **MapInfo_Databases\Databases** directory and click on **Properties** in the pop-up box
	Step 3 Under **Attributes**, ensure that **Read Only** is NOT checked
	Step 4 Click **Apply**
	Step 5 Check **Apply changes to this folder, subfolders and files** and click **OK**.
	Step 6 Click **OK**
Task 2 Check your data's file types	**Step 1** Browse to the data in **MapInfo_Databases\Databases**
	Step 2 Click on **View	Details** in your file browser window. This will show an overview of the directory contents, including the file type (in *Vista* you may need to go to **View | Choose Details** and select **Type**). The file type determines how it can be imported in *MapInfo*. Note that one of the files is text and another is an Excel spreadsheet. The others are in native *MapInfo* formats.
	Step 3 Close this window when you are done

Task 3 Start *Excel* and open the spreadsheet	**Step 1** From the **Novell-delivered Applications** window, click on the *Excel* icon. *Excel* will start.
	Step 2 Click on **File | Open**, browse to the folder **MapInfo_Databases\Databases**, and select the file **Sites_EnglandWales_1**. Click **Open**. The spreadsheet will open
	Step 3 Note the structure of the database: what the columns (fields) are called, and what kinds of values (data types) they contain, *e.g.* integers, floats, text, dates, etc. *MapInfo* will attempt to automatically recognise these when it opens the file. Also, note the fact that the first row of the database doesn't in fact contain any data at all – it contains the column titles. This is important. Lastly, note the columns containing the coordinates, called **NatGrid_Easting** and **NatGrid_Northing**. Knowing the coordinate system your coordinates are referenced to is absolutely essential. In this case, it's a safe bet that 'NatGrid' refers to the UK National Grid.
	Step 4 Exit *Excel*.
Task 4 Start *Notepad* and open the text file.	**Step 1** From the **Novell-delivered Applications** window, click on the *Notepad* icon. *Notepad* will start.
	Step 2 Click on **File | Open**, browse to the folder **MapInfo_Databases\Databases**, and select the file **Sites_ScotlandNIreland_1**. Click **Open**. The text file will open
	Step 3 Note the structure of the database. This is specifically a *tab-delimited* text file. The cells in the database are delimited (separated) by tab characters. The point of delimiting is that whatever application later opens this file, it will know that the chosen delimiter (tab) signifies the start of a new cell. Another popular delimiter is the comma, hence you can have comma-delimited files. Sometimes the forward slash is used to delimit. Technically any character can be used as a delimiter, but of course it has to be a character that doesn't appear in the contents of your database. Note that there are five columns, and that these are similar to the *Excel* spreadsheet. Take notice, however, of the coordinates. These are **DD_Longitude** and **DD_Latitude**. In this case, DD stands for 'degrees decimal,' as opposed to DMS for 'degrees/minutes/seconds.' This basically just means that the longitude and latitude coordinates have been converted into decimals.

	Step 4 Exit *Notepad*.
Task 5 Start *MapInfo*	**Step 1** From the GIS directory of the **Novell-delivered Applications** window, click on the *MapInfo* icon. The *MapInfo* workspace window will open and the **Quick Start** window will appear, giving you several options.
	Step 2 In **Quick Start**, click **Cancel**. By doing so you will effectively enter an empty workspace. The only option on **Quick Start** that really saves any time is the option to re-open your last workspace.
Task 6 Open your vector base map	**Step 1** Click on **File | Open,** browse to **MapInfo_Databases\Databases** and open the table file **UKIreland_Admin_poly**. A map window will open showing polygons of the British Isles.
Task 7 Maximise the map window	**Step 1** Click on the 'maximise' icon in the upper-right corner of the inset map window, so that it fills the workspace. Zoom out so that you can see most of the British Isles.
Task 8 Open the *Excel* spreadsheet	**Step 1** Click on **File | Open,** change **Files of Type** to **Microsoft Excel (*.xls),** browse to **MapInfo_Databases\Databases** and open the table file **Sites_EnglandWales_1**. An **Excel Information** window will open.
	Step 2 Enter the parameters for the import process. The **Named Range** is the portion of the spreadsheet that you want to import. The default is to import the entire worksheet – next to **Current Value** you will see a range of cells, such as A1:E166, meaning that everything from cell A1 to cell E166 will be imported. You have the option to change this, but leave it as it is. Enable the option to **Use Row Above Selected Range for Column Titles**. You will do this whenever the first row of your database contains the titles of the columns (fields). Notice what happens to the **Current Value** when you do this: A1 will change to A2. This means that, instead of starting at Row 1, it will start importing content at Row 2. This is exactly what you want it to do. Click **OK**. A **Set Field Properties** window will open.

	Step 3 In **Set Field Properties**, *MapInfo* is attempting to automatically recognise the data types in your spreadsheet. It usually does a pretty good job, but there are a few things you need to check. Make sure that your types are correct – integers are whole numbers only, floats are numbers that may include decimals, characters are for text, etc. Highlight the field **Name**, and notice that a **Width** parameter appears. The is the maximum length of any entry in this field. *MapInfo* will automatically detect the longest entry you have, and set the width to this. If you want to be able to later add longer text entries, you will need to change this width now. If you later try to type text into a field that is too short for it, *MapInfo* will simply truncate your entry. After the table has been created, is possible to later change the length of this field (**Table | Maintenance | Table Structure**). For now, just click **OK**. The **Set Field Properties** window will close, and a browser will appear, showing the results of the import.
	Step 4 Look over the table and see that it imported correctly. The first site should be ASHINGTON. When the browser window is active (*i.e.* the bar along the top of the window is blue, or in this case, where the browser window is maximised), look at the lower-left corner of the workspace. You will see a small readout displaying something like '**records 17 - 32 of 165.**' This is telling you the range of the database that is visible in the browser: records 17 to 32 out of 165 total records. If you recall, there were 166 records in the Excel spreadsheet, but the first record contained no content. So that's as it should be.
Task 9 Open the text file	**Step 1** Click on **File | Open,** change **Files of Type** to **Delimited ASCII (*.txt),** browse to **MapInfo_Databases\Databases** and open the table file **Sites_ScotlandNIreland_1**. A **Delimited ASCII Information** window will open.
	Step 2 Enter the parameters for the import process. By default, the delimiter is Tab. This is what your file uses, so leave it like that, but note that you could change it to a comma or whatever you like. Leave **File Character Set** as it is. Enable the option to **Use First Line for Column Titles**. This is just like the last table. Click **OK**. A browser will open immediately.

	Step 3 Note that, this time, there was no **Set Field Properties** window. This is simply how *MapInfo* imports delimited text. It will try to recognise the data types automatically, but it will not give you the option to set them yourself, as it did with the *Excel*. In fact, in this case it will actually interpret the data types in the text file differently to the way it interpreted the *Excel*. If you go to **Table | Maintenance | Table Structure** and check the structure of each of the two imported tables, you can see that in the ScotlandNIreland table the Site_ID and Type fields were interpreted as small integers, and in the EnglandWales table they were interpreted as floats. This won't make any difference to what you do in this session, and it's unlikely to ever cause problems in your own work, but it's worth checking this.
	Step 4 Look over the table and see that it imported correctly. The first site should be Kirkwall. There should be 42 records.
Task 10 Save copies of your files, close the originals, and open the copies.	**Step 1** Go to **File | Save Copy As** and save a copy of **Sites_EnglandWales_1**, and name it **Sites_EnglandWales_2**. The reason for doing this is a little complex, and it is not necessarily always desirable. When you imported the files, *MapInfo* didn't actually translate the entire data content of them. All it did was create a small *table definition* file, that merely tells *MapInfo* how to interpret the *Excel* and text files, and where they are. The next time you try to open those files in *MapInfo*, it will simply repeat the import process, but without you having to enter the details. The problem here is that, if in the meantime you make any changes to your *Excel* file, text file, or whatever the source file is, when *MapInfo* next opens it, it may generate errors, or not import the entire document. The way around this is to save a copy of the table. When you do this, the second version (2) will be saved in *MapInfo*'s native format. You will effectively be forcing *MapInfo* to translate everything.
	Step 2 Save a copy of **Sites_ScotlandNIreland_1** as **Sites_ScotlandNIreland_2**.
	Step 3 Close **Sites_EnglandWales_1** and **Sites_ScotlandNIreland_1 (File | Close Table)**

	Step 4 Open **Sites_EnglandWales_2** and **ScotlandNIreland_2.**
Task 11 Leave your workspace open. Save the workspace if you like, but since you have only a few files open, there isn't really much point.	

4 Creating Databases from Scratch in MapInfo

It is entirely possible to create your database from scratch from within *MapInfo*. If you take time to experiment with this, you will find the *MapInfo* interface a little "clunky" for inputting data, and it certainly doesn't have all of the functionality of a dedicated program. However, if you just have a relatively small amount of data, you may find it more convenient than importing.

You will need to decide right away what the table is to be named, and how it will be structured, *i.e.* column titles, data types, etc. Once the table is created and opened in a browser, it's really just a matter of adding rows to it and entering the values into each cell.

Exercise 2 Creating Databases from Scratch in MapInfo

- *In this exercise, you will practice creating a small database within MapInfo and entering data into it. You will then use this table in a later exercise.*

Task 1 Create a new table	**Step 1** Click on **File	New Table,** or simply press CTRL + N. A **New Table** window will open.
	Step 2 Under **Create New Table** you have several options for whether you want to open a new browser window for the table, a new map window, and/or add it to an existing map window. Enable **Open New Browser**. Disable both map window options.
	Step 3 Under **Table Structure** you have the options of either creating the structure of the table from scratch, or basing its structure on another table. The structure, by the way, is the number of columns in the table, the title of each column, and the data type each column contains. The **Using Table** option is very useful if you want to create another table in the same format as one you already have. However, in this case we don't have one, so select **Create New**. Click **Create**. A **New Table Structure** window will open.
	Step 4 The **New Table Structure** window will be vaguely familiar. It's similar to the **Set Field Properties** window you encountered in Exercise 1. As you might guess, this is where you define the table's structure. To start, click **Add Field**. A new field will be added to the list, and that field's properties will populate the **Field Information** in the lower-left corner of the window.
	Step 5 Enter the field properties for the first field. In this case, we want the first field to be named **UK_Region**, we want its type to be **Character**, and its width to be **20** characters. Note that we must use underscore instead of a space, because *MapInfo* will not allow any spaces in the column titles (spaces in the table's content are okay, however). When this is done, click **Add Field** to add another field.

	Step 6 Name the second field **Count** and set the type to **Integer**. Then add another field.
	Step 7 Name the third field **EUADMLL_ID** and set the type to **Integer**. Note that the fields have been added to the list one after the other. This is also the order (from left to right) in which the columns will appear in the table, *i.e.* the top field in the list will appear on the left of the table. If you care about the order in which your columns appear in the table, re-order them here, using the **Up** and **Down** buttons. You can also drag and drop the columns later, once the table is open.
	Step 8 Select the projection. Press **Projection**, which will open a **Choose Projection** window. This should be familiar. The projection contains information about the coordinate system referenced by any coordinates in the table, and information about how to plot them onto a flat map surface. In this case, since our study area is in the UK, the most sensible projection to use is the British National Grid (Category = British Coordinate Systems, Member = British National Grid). Then click **OK** to return to the **New Table Structure** window.
	Step 9 Check over your field properties to make sure that they're the way you want them. You can always change them later by going to **Table | Maintenance | Table Structure.** When you're ready, click **Create**. A **Create New Table** window will open.
	Step 10 Browse to a location to put the new table, and give it a name. In this case, put it in the **MapInfo_Databases\Databases** folder and name it **Scotland_Regions**. Then click **Save**. *MapInfo* will create the new table file immediately in the specified location, and it will open a new browser window. At first, the browser will show only the column titles. The table doesn't contain any records (rows) yet.
Task 2 Add records to the table, and populate them with data	**Step 1** Click on the browser window to make it active, if it isn't already. The top of the window will turn from grey to blue. If your browser window is maximised in the workspace, you don't need to worry about this.
	Step 2 Add a new row (record) to the table by pressing CTRL + E.
	Step 3 Click on the first cell in the record, under UK_Region. The cell's border will become highlighted. Type the text **`NW Scotland`**. When you're done typing, click somewhere outside of the cell to enter the data.
	Step 4 In the **Count** field, enter the integer **`482`**. Under EUADMLL_ID, enter **`1072`**.

	Step 5 Add a new row (CTRL +E) and input the following data: UK_Region = NE Scotland Count = 629 EUADMLL_ID = 1233
	Step 6 Add a new row (CTRL +E) and input the following data: UK_Region = SE Scotland Count = 849 EUADMLL_ID = 1312
	Step 7 Add a new row (CTRL +E) and input the following data: UK_Region = SW Scotland Count = 367 EUADMLL_ID = 1327
	Step 8 Since you've made changes to the table's content, you should save the table (**File | Save Table**). No need to save the workspace at this point.
	Step 9 Close the browser windows displaying your tables. Leave the map window open.

5 Updating and Querying Databases Using Expressions

At the moment, the two tables you imported in Exercise 1 and the table you created from scratch in Exercise 2 contain records, but no corresponding map objects. You will create these objects in Exercises 4 and 5. Before that, it is worth having a look at how to use expressions to query and update your tables.

MapInfo expressions, and specifically the language and syntax used to write them, are a bit difficult to get your head around at first. However, once you do figure them out, you will find them to be a very powerful way of exploring, analysing, and editing your data.

There are two main uses for expressions in *MapInfo*. The first is updating a field (column) in a database. In this case, the expression defines the new value that will go into that field – if you have ever used expressions in a DBMS (e.g. *Excel*) this will be familiar. For the most part, these expressions are mathematical and result in a numerical output. However, it is also possible to process text. For example, if you have two text columns you can 'add' them together to create a third column, in which they're appended. For instance, if you had the first two columns in the following table:

City_Name	State_Name_Abbr	City_and_State_Name
Roanoke	VA	Roanoke (VA)
Memphis	TN	Memphis (TN)

you could then generate the third column using the expression:

```
City_Name + " (" + State_Name_Abbr + ")"
```

It may look unusual, but if you read through the expression as though it were an instruction to a person, it may be easier to understand:

'City name, then a space and an open parentheses character, then State name abbreviation, then a close parentheses'

The other main use for expressions in *MapInfo* is querying a table. Note that, when you query a table in *MapInfo*, the results are stored in the Selection, which acts a bit like the Clipboard in other programs. You can have only one Selection at a time, but you can do most anything you like to it – save a copy of it, zoom to it, move it, delete it, and so on. You can of course also view a selection in a browser window and calculate statistics from it, so it allows you to search through the data and analyse it.

There are two tools for running queries in *MapInfo*. The first simply uses *MapInfo* expressions, and the second uses Structured Query Language (SQL) expressions, which is a bit more sophisticated. However, the basic idea is the same in both: the expression you write defines the conditions that a record must satisfy in order to be selected. *MapInfo* then applies these conditions to each of the records in the table and determines which ones 'pass.'

Expressions do have other uses in *MapInfo*, *e.g.* for creating labels for a layer, but these are perhaps less common.

Exercise 3 Updating and Querying Databases Using Expressions

- *In this exercise, you will practice updating a database column using a MapInfo expression. Specifically, you will calculate population density for areas based on their total population and geographic area. You will also practice querying a database by writing a MapInfo expression that defines pass/fail conditions, applying it to a database, and browsing the results.*

Task 1 Open **UKIreland_Admn_poly** and display it in a browser window	**Step 1** If it's not still open in the workspace, click on **File | Open,** and browse for it. This table contains map objects, so when you open it it will by default open in a map window. Also, chances are that the map window will be zoomed in to the centre of the data coverage. Keep the map window open, but zoom out until you have a view of the entire UK.
	Step 2 Open a new browser window displaying **UKIreland_Admn_poly** by clicking on **Window | New Browser Window**.
	Step 3 Note what columns are included in this table. It contains population statistics for the UK by administrative region for four years (1987-90) and total areas.
Task 2 Create a new column in **UKIreland_Admn_poly** called **POP_DENSITY_87_SQKM**.	**Step 1** Click on **Table | Maintenance | Table Structure**. Select **UKIreland_Admn_poly** and click **OK**. A **Modify Table Structure** window will open.
	Step 2 Click on **Add Field**, and create a new field called POP_DENSITY_87_SQKM. Set its data type to Float.
	Step 3 Click **OK**. *MapInfo* will change the structure of **UKIreland_Admn_poly** and then remove it from any maps or browsers that are displaying it. However, it is still open in the workspace. Open a new map (**Window | New Map**) and a new browser (**Window | New Browser**) showing it.
	Step 4 In the browser window, scroll to the right to confirm that the new column has been added to the table.

Task 3 Update the new column using an expression to calculate the population density.	**Step 1** Click on **Table	Update Column**. An **Update Column** window will open.
	Step 2 Enter the parameters for the column update. In **Table to Update**, choose **UKIreland_Admn_poly**. In **Column to Update**, choose **POP_DENSITY_87_SQKM**. In **Get Value from Table**, choose **UKIreland_Admn_poly**. In the Value field, you can write the expression used to calculated the updated value. However, it is much easier if you click on **Assist**. This will open another window that helps you write the expression, and when you are done there, it will plug it into this field. So, click on **Assist**. An **Expression** window will open.
	Step 3 The text box to the left of the window displays the expression you are writing. You can type directly into this box, or you can use the menus on the right of the window to insert terms into the expression, such as columns, operators, and functions. Drag down the **Columns** menu and select **POP87**. Then drag down the **Operators** menu and select the forward slash (this represents division). Then drag down the **Columns** menu and select **TOTAREA**. If you have done this correctly, in the text box you will see: **POP87 / TOTAREA**. This essentially means, 'divide the column POP87 by the column TOTAREA.' This of course is how you calculate population density.
	Step 4 Click **Verify**. *MapInfo* will examine the expression you have written and tell you whether the syntax is correct. Note that this determines only whether *MapInfo* can understand your expression – it doesn't guarantee that the expression you have written will do what you intend it to do! If your syntax is correct, *MapInfo* will tell you so. Click **OK** to return to the **Expression** window, then **OK** to insert the expression back into the **Update Column** window. Note that the **Value** field has now been filled.

	Step 5 If you want to open a new browser window displaying the results, then enable the **Browse Results** option. In this case, do so. Then click **OK**. *MapInfo* will run the calculation to update the column. It will display the results in a new browser window. Confirm that the new column has been populated with the results.
Task 4 Save **UKIreland_Admn_poly**. Because you have changed the table's contents, you should save it (**File | Save Table**)	
Task 5 Select the records in **UKIreland_Admn_poly** where the 1987 population density is greater than 200 people per square kilometre, by writing a query for the **POP_DENSITY_87_SQKM** column.	**Step 1** Click on **Query | Select**. A **Select** window will open. It is perhaps useful to think of this process as just another way of selecting records – instead of doing it graphically through a map or browser window, you are doing it through an expression.
	Step 2 Enter the parameters for the query. **In Select Records from Table**, choose **UKIreland_Admn_poly** . Under this, there is a text box labelled **That Satisfy**. These are the conditions that a record must satisfy in order to be selected by the query. As you might guess, an expression goes into this box, and there is an **Assist** button to help. Click on **Assist**. An **Expression** window will open.
	Step 3 This expression window is the same as the one in Task 3. This time, enter the expression: POP_DENSITY_87_SQKM > 200 Under this condition, all of the records in the table where the population density is greater than 200 will be selected. Use **Verify** to check that your syntax is correct, then click **OK** to go back to the **Select** window.

	Step 4 Enter the remaining parameters. In **Store Results in Table**, leave this as **Selection**. If you wish to sort the results by column, there is an option to do so. Leave the **Browse Results** option enabled. Note that there is an option to **Save Template**. This will save the parameters for your query, which is useful if you run several similar queries, or if you have a query that is complicated. In this case, however, the query is pretty simple, and we're only going to run it once. Click **OK**. The query will run, and a new browser window will open displaying the results.
Task 6 Look over the results of the query, then unselect them	**Step 1** In the new browser window, scroll to the right to view the population density column, then scroll down. In the selected records, there should be a population density greater than 200 people per square kilometre.
	Step 2 Look in any map window displaying **UKIreland_Admn_poly**. Several of the regions should be selected (highlighted). These are the same records that were selected by the query.
	Step 3 Click on **Query | Unselect All** to unselect the results of the query. However, note that you can take a bit of a shortcut back to the query results – if you go to **Window | New Browser Window**, you will see that **Query 1** appears just as if it were a table. However, this Query 1 is not currently saved as a file.

6 Projecting Coordinate Data and Creating Map Objects

The two tables you imported in Exercise 1 contain coordinate data. In *MapInfo* it is quite simple to project (*i.e.* plot) these and create objects as points. However, it is absolutely critical that you know the coordinate system that the coordinates are based on.

There is an important distinction to make between coordinate systems and map projections. The coordinate system is the system for expressing locations in space. In Cartesian coordinate systems, these take the form of (X,Y) or (Easting, Northing) coordinates, and in Spherical systems they are (Longitude, Latitude). The coordinate system is the frame of reference – think of it as the *language* in which the coordinates are written.

The map projection, on the other hand, is the system for plotting that coordinate data onto a 2D surface, *i.e.* as a map. The projection is superficial, in the sense that the same set of coordinates can be plotted using any map projection, and the location may be 'correct' in each. Think of it as the visual *style* in which the data is presented. Take note, however, that all map projections of the Earth's surface distort it to some extent. This is because maps try to represent a curved surface using a flat surface.

Unfortunately, *MapInfo*'s interface is perhaps a little confusing on this point. It uses the term 'Projection' to refer to both the coordinate system and the map projection.

Exercise 4 Projecting Coordinate Data and Creating Map Objects

- *In this exercise, you will practice projecting coordinate data and creating map objects as points.*

Task 1

If **UKIreland_Admn_poly**, **Sites_EnglandWales_2**, and **Sites_ScotlandNIreland_2** are not open, then open them to the workspace. Display **UKIreland_Admn_poly** in a map window (this will be the base map). Open **Sites_EnglandWales_2** and **Sites_ScotlandNIreland_2** in browser windows.

Task 2	**Step 1**
Create points for **Sites_EnglandWales_2** and add it	Click on **Table	Create Points**. A **Create Points** window will open.

to the map window.	**Step 2** Enter the parameters for creating the points. In **Create Points for Table**, select **Sites_EnglandWales_2**. Next to **Using Symbol,** you can see a preview of the point symbol that will be used as the default style for the created points. You can change it here (by clicking on the icon), but of course you can always override this style later through **Layer Control**. In this case, leave the symbol as it is. In **Get X Coordinates from Column**, select **NatGrid_Easting**, and in **Get Y Coordinates from Column**, select **NatGrid_Northing**. Ignore the two **Multiply** fields. Note that you have the option to **Overwrite existing points**. If you leave this disabled, you will not overwrite any points that were previously created for this layer. If you enable it, then each time you run the tool you will overwrite any points that were previously created for this layer (*e.g.* if you had made an error in creating points the first time, or had added records, and wanted to create the points again). Since this is the first time you have run the tool on this layer, it is irrelevant whether the option is enabled.
	Step 3 Click on **Projection** and choose the projection. In this case, what you are actually doing is choosing the coordinate system the coordinates are based on. In this case, you happen to know that the coordinates refer to the British National Grid. Choose **Category = British Coordinate Systems**, and **Category Member = British National Grid**. Then click **OK** to return to the **Create Points** window.
	Step 4 Click **OK**. The **Create Points** window will close. Go to the map window displaying **UKIreland_Admn_poly**, open **Layer Control**, and add **Sites_EnglandWales_2** to the map. A set of points should appear on the map, covering England and Wales.

Task 3

Create points for **Sites_ScotlandNIreland_2** and add it to the map window.

You will follow the exact same process an in Task 3 above *except* you will use the following parameters:

Create Points for Table = Sites_ScotlandNIreland_2

Get X Coordinates from Column = DD_Longitude

Get Y Coordinates from Column = DD_Latitude

Projection Category = Latitude / Longitude

Projection Category Member = Latitude / Longitude

As you can see, this time the coordinates are based on Latitude/Longitude, not the UK National Grid. If you create them correctly, your points will end up covering Scotland and Northern Ireland.

7 Associating Database Records with Existing Map Objects

In the previous section, you created point map objects from coordinate data. What do you do if you don't have coordinates in your database, but only references to geographic features? In this case, you will need to link your table to a table that already contains these features as map objects – often, but not always, this will be an off-the-shelf dataset. For this to work, you will need to have a column that is the same in both tables, to act as an index. What you will then do is append columns from your table, one at a time, onto the table containing the map objects.

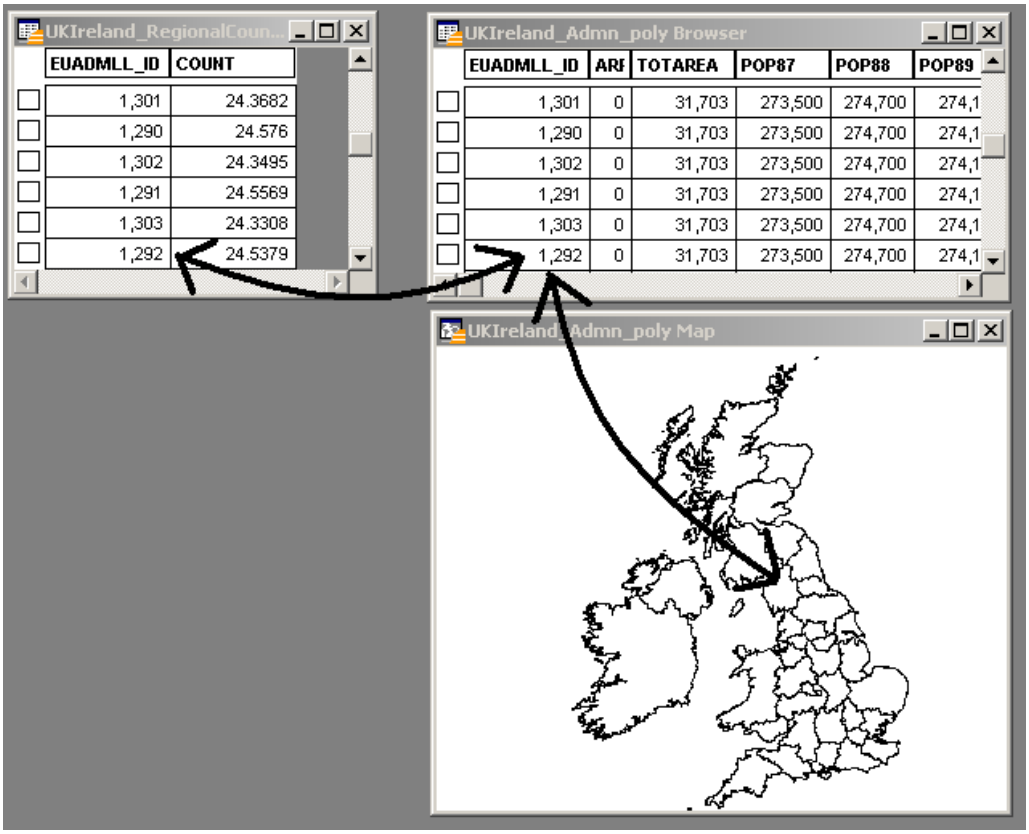

Figure 6 If two tables contain a common field (in this case, EUADMLL_ID), that field can be used to join the tables. For example, the Count field from the table on the left can be appended onto the table on the right, which contains map objects

Exercise 5 Associating Database Records with Existing Map Objects

- *In this exercise, you will practice joining one table to another. This is a useful way of relating your data, in one table, to geographic features represented as map objects in another table*

Task 1

Open **UKIreland_Admn_poly** and **UKIreland_RegionalCounts** to the workspace, if they are not already open. If your workspace is becoming cluttered, close or minimise any other browsers or maps.

Task 2

Create a new column for **UKIreland_Admn_poly** (**Table | Maintenance | Table Structure**). Name the new column **RegionalCount**, and make its data type **Float**. Click **OK**.

Task 3

Update the new column in **UKIreland_Admn_poly**

Step 1

Click on **Table | Update Column**. An **Update Column** window will open.

Step 2

Enter the following parameters:

Table to Update = UKIreland_Admn_poly

Column to Update = RegionalCount

Get Value from Table = UKIreland_RegionalCounts

Note that when you entered **UKIreland_RegionalCounts** in the **Get Value from Table** field, the **Join** button became active. *MapInfo* recognises that, since you're trying to update a field in one table from a field in another table, you must be trying to join then. Click **Join**. A **Specify Join** window will open.

Step 3

In the **Specify Join** window there are two options for creating the join. The first is a descriptive join, *i.e.* relating a field in one table to a field in another table. The second is geographic, allowing you to join tables based on how objects in one table contain, are contained in, or intersect with those in the other. In this case, you want the first option. In the first menu for this option, select the **EUADMLL_ID** field, and in the second menu, select **EUADMLL_ID**. When this is done, the specification should read:

'Join where EUADMLL_ID from table UKIreland_Admn_poly matches EUADMLL_ID from table UKIreland_RegionalCounts'

Click **OK** to return to the **Update Column** window.

	Step 4 In the **Calculate** field, select **Value**. This means that the values will simply be transferred from one table to another. However, there are also options for analysing data in the process. In the **of** field, select **COUNT**. Click **OK** to run the update. A new browser will open, showing the results.
	Step 5 Browse the new results and confirm that the **Regional Count** column in **UKIreland_Admn_poly** has been populated using data from **UKIreland_RegionalCounts**.
Task 4 Save the **UKIreland_Admn_poly** table.	

8 Thematising Map Objects Using Colour-coding and Symbology

One of the core functions of *MapInfo* is generating thematic maps. It is possible to modify the appearance of map objects systematically, according to any of the fields in their corresponding records.

There are several types of thematic maps in *MapInfo* and for each of these types there are variants for thematising point, line, and polygon features. The easiest way to learn about these is to explore the menus used to create and edit them.

MapInfo also has the ability to generate size-graduated symbols, dot density maps, bar charts, and pie charts.

For most thematic maps, you have the option of creating a Legend window, which can be customised and then saved as an image.

As you create thematic maps, you will notice that they are added to the Layer Control as though they were tables. You will notice several things about them. They are always added above the layer from which they were generated, their visibility can be toggled on and off, and you can create as many thematic maps as you want from any given layer of objects. There is also a **Thematic** button on Layer Control that allows you to edit thematic map layers.

Exercise 6 Thematising Map Objects Using Colour-coding and Symbology

- *In this exercise, you will practice making two common types of thematic maps: a colour-coded map representing a numerical field (in this case, population of the UK by region) and a symbolised map representing a categorical field (in this case, site types).*

Task 1

Open **UKIreland_Admn_poly**, **Sites_EnglandWales_2**, and **Sites_ScotlandNIreland_2** and add them together to a single map window. Open one browser for each. Close any other windows.

Task 2	**Step 1**
Create a thematic map for **UKIreland_Admn_poly** representing the 1987 population density field as colour ranges.	Click on the map window to make it active. The map window needs to be active in order for there to be a **Map** item in the workspace menu bar. Click on **Map | Create Thematic Map**. A **Create Thematic Map (Step 1 of 3)** window will open.

	Step 2 Below **Type**, on the left of the window, you will see a number of icons. Each of these is a thematic map type. Try clicking on each one, and observe the many templates available for each. Highlight each template and have a look at the preview window. The preview shows a sample of what that template might look like. When you are done exploring, under **Type** choose **Ranges**, and under **Templates**, choose Region **Ranges, Solid Red, Dark-Light**. Then, click **Next**. The second page (2 of 3) will open.
	Step 3 On page 2 of 3, you will select the table and field to base the theme on. In this case, set the table to **UKIreland_Admn_poly** and the field to **POP_DENSITY_87_SQKM**. There is an option to ignore zeroes or blanks, which might be useful if there are gaps in your data. In this case, it is irrelevant. Click **Next** to go to the third page.
	Step 4 On page 3 you will get a preview of the thematic mapping scheme. In this case, the preview will show you several ranges of values for POP_DENSITY_87_SQKM, and the display style to be applied for each (here, the styles are different shades of red). The number in parentheses following each range is the number of records in the table that fall into that category. You will also notice buttons in the upper-right corner that allow you to edit the ranges, styles, and legend window. Click on any of these and you will be open new windows where you can change the settings of the thematic map. Of course, you can always change them later, through the **Thematic** button on Layer Control. Make a few slight changes if you wish. When you are ready, click **OK**. *MapInfo* will create the thematic map layer and add it to the map window.
Task 3 View and edit the thematic map for **UKIreland_Admn_poly**.	**Step 1** Look at the thematic map in the map window. Notice how the thematic style has been applied to the map objects.

	Step 2 Open the **Layer Control** and observe that the thematic layer has been added above the **UKIreland_Admn_poly** layer. Note that it is slightly indented. This is to signify that, like all thematic layers, it is dependent on the layer on which it is based. If you were to switch off the visibility of the **UKIreland_Admn_poly** layer, *MapInfo* would ask you if you want to switch off its thematic layers too. If you move **UKIreland_Admn_poly** up or down in the drawing order, the thematic layers will move with it. If you remove **UKIreland_Admn_poly** from the map or close it from the workspace, the same thing happens to its thematic layers. The thematic 'layer' doesn't have any content – it's just a way of viewing the **UKIreland_Admn_poly** layer. If you have a thematic layer that you like, be sure to save the workspace, because that is where it will be stored. Also, if you plan on making more thematic maps in the same style, you can save a template for it and load it the next time you want it.
	Step 3 With the **Layer Control** still open, highlight the thematic layer and click on **Thematic.** A **Modify Thematic Map** window will open, which is identical to page 3 from the map creation process. Make any changes you want here, then click **OK** to go back to **Layer Control** and **OK** to apply the changes to the map.

Task 4 Create a thematic map for **Sites_EnglandWales_2** representing the Type field as symbols.	**Step 1** This is the same process as Task 2 above *except* in this case, use the following parameters: Page 1 **Type = Individual** **Template = Point, IndValue, Black and White** **Next** Page 2 **Table = Sites_EnglandWales_2** **Field = Type** **Next** Page 3 (no changes, use defaults) **OK** The **Sites_EnglandWales_2** layer should now appear as black squares, circles, and diamonds.