Making first base

Tim Anderson takes a look at MSDE, Microsoft's client/server database engine that is free to deploy.

Most real-world applications use databases, so a key decision early in the life of a project is which database to use. At one extreme you can roll your own using low-level file manipulation, and at the other you might want to connect to a mainframe.

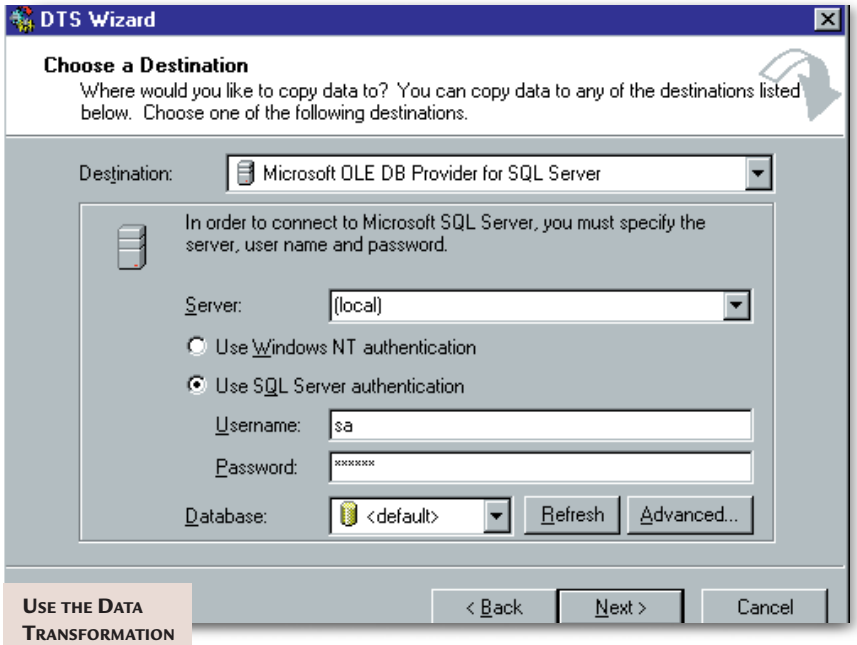
Small systems built with Visual Basic tend to use JET, the database engine of Microsoft Access, and the parallel choice for Delphi is usually Paradox. Both are excellent database engines, and for developers wanting to deploy applications they have the huge advantage of being free to distribute.

Since the arrival of Office 2000, Microsoft has offered another option. The Microsoft Database Engine (MSDE) is also freely distributable by licensed users of Office Developer Edition or Visual Studio 6.0.

By contrast, similar products from Inprise (Interbase) and Sybase (SQL Anywhere and its derivatives) do require licences, so this is a good deal. MSDE itself is really SQL Server with a few restrictions, but the full SQL Server is expensive to deploy.

But why move from comfortable old JET to the unknown territory of MSDE? Well, there are several reasons. First, JET works through file-sharing while MSDE is client/server. It is important to be clear about the difference.

Imagine a network with two workstations and a shared Access database. To connect to the data, each workstation loads its own independent copy of JET into memory and opens the database file across the network.



USE THE DATA TRANSFORMATION SERVICES WIZARD TO IMPORT DATA TO MSDE

JET might not even be installed on the server where the data is located.

This makes it difficult to prevent large amounts of network traffic, as the database file is read by the application. It also means that because each instance of JET runs in isolation, it is difficult to optimise performance when there is multi-user activity.

Finally, the integrity of the data depends on the proper refreshing of the database file as seen by the application, when its contents change because of other network users writing and deleting data.

This last point is particularly aimed at NT users. A common problem is database corruption caused by over-enthusiastic caching of data on the server, when there is heavy activity from several users of the same file. Novell is reportedly much better in this respect, although NT can be tuned to prevent most problems.

File-sharing database engines are only suitable for small numbers of users

The bottom line is that file-sharing database engines are only suitable for small numbers of users. For JET, Microsoft suggests a maximum of 20, and fewer than that if the system is to be used intensively. There is a built-in limit of 255.

In contrast, MSDE is a client/server database. The database file is only accessed by one instance of the database engine, running on the server. Small client libraries run on each workstation, to handle the connection to the database engine.

MSDE does all the processing of queries and updates on the server, and feeds back results to the clients. The system is inherently more suitable for multiple users, partly because network traffic is less, and partly because MSDE can be smart about handling multiple connections.

The full SQL Server has no specific limit on the number of concurrent users, and if performance is a problem it is easy to upgrade the server. MSDE has additional features that JET lacks, such as transaction logging, which lets you backtrack and recover the database as it was at a previous moment in time.

Should you use MSDE?

There are a couple of points against it. One is that it is larger, so if you are

deploying a single-user application the system requirements are greater. Another is that it is different and more complex, so experienced JET developers have some learning to do. Installation is also more difficult. Otherwise, the arguments in favour are compelling.

How to move to MSDE

Developers have to know more than how to simply read and write data. There is also the matter of how to design the database, create the tables, establish security and enforce relational integrity.

The full version of SQL Server comes with Enterprise Manager, but not with MSDE. The twist is that Visual Studio 6.0 comes with a developer version of SQL Server, so that you can use Enterprise Manager to manage databases, and then deploy them using MSDE.

The second twist is that Access 2000 can connect natively to MSDE and provides visual tools to manage the data. Note that MSDE is not installed by default. Find the SQL directory on the Office 2000 CD, and run SETUPSQL.EXE from the x86/setup directory.

If you have Visual Studio 6.0, you can download the setup package MSDEX86.EXE from Microsoft's website. This installs the database engine along with the service manager for starting and stopping it, but not much else.

It is not intended that you use this for development, but it is essential for deploying MSDE applications, and comes with an installation script that you can modify. It works on Windows 95, 98 and NT.

The service manager is an important clue as to how MSDE actually works. A service is an application that runs in the background. On Windows NT you can set a service to run even when no user has logged onto the machine. The service has to be running for you to access the data.

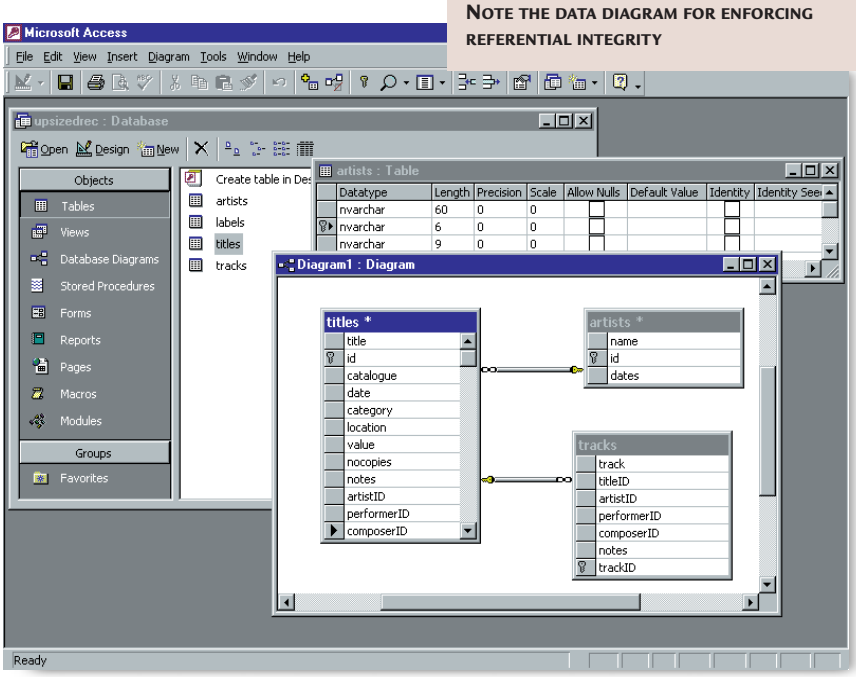
The service manager is an important clue as to how MSDE actually works

To add a database to MSDE, the data has to be created or imported under the control of a specific MSDE installation. In other words, you cannot simply copy a database file into a directory – it has to go through the database engine.

This makes deploying an application a little trickier, but has the advantage that you do not need to worry where the data is stored.

All that you really need to know is which MSDE server is actually managing the database that you want to access.

In Access 2000, using MSDE is just a matter of creating a Data Project. Note the data diagram for enforcing referential integrity



Getting started

Once you have installed either MSDE for Office 2000 or some other version of SQL Server 7.0, you can create a database. One easy way is to use Data Transformation Services (DTS) to import an existing Access (or other) database.

The Office 2000 installation provides an Import and Export (DTS) wizard that turns up in the MSDE group on the Start menu, putting a friendly face on DTS. There is also an Upsizing Wizard in Access, which is more fully automated but gives less control.

Note that with SQL Server you will always be asked for a user name and password, which by default is 'sa' with a blank password, unless you are running Windows NT and using integrated authentication.

It is worth exploring the DTS wizard, as it offers a wide range of options. For example, you might want to change a column (field) name, or amend the wizard's decision about the correct SQL Server field type for a particular field.

Normally, you will also want to change the database destination from 'default' to 'new', unless you really want to add tables to an existing database.

A few points to consider are that character field types come in Standard and Unicode types, the latter shown by an initial 'n' in the type name, as in char and nchar. The wizard defaults to Unicode types, but these take up twice as much space and could be an unnecessary indulgence if you do not need the extra characters.

Next, note that some fields are prefixed 'var', which means variable length. This affects the way the data is stored, with fixed-length fields being better for performance but variable length making better use of space.

If you import a database, you should take a careful look at the results. You can do this with either Access 2000, by creating a Data Project based on the new database, or by using SQL Server Enterprise Manager if you have SQL Server 7.0.

Depending on the method used (DTS, Upsizing wizard) and the options

(FIG 1)

VBA routine to show a dialog from a contact form

```
Sub showdialog()
Dim thisitem as ContactItem
If TypeName(Application.ActiveInspector.currentItem) <> "ContactItem" Then
Exit Sub
End If
Set thisitem = Application.ActiveInspector.CurrentItem
' ... use reference to Contact to set values in the dialog and save changes
UserForm1.Show vbModal
Exit Sub
```

chosen, you may still have considerable work to do. Check that there is a primary key defined for each table, otherwise the database will not work properly. You should also check other indexes and referential integrity rules, since some import procedures carry these over imperfectly or not at all.

If you use AutoNumber fields, check that these have translated to SQL Server Identity columns. Another point to check is whether you want to allow null values in each field. Needless to say, it would be unwise to take a business-critical application, upsize the data to MSDE and run live against the new database immediately.

Using MSDE

Once you have an MSDE database up and running, you can start building applications that use the data. SQL Server has been around for some years and there are several ways to control it via programming.

An Access 2000 data project lets you work with MSDE much as if it were JET, the most significant difference being that whereas a JET MDB generally contains the data itself, a data project only stores forms, code and other application elements.

To get at the data from Visual Basic, Delphi or other languages, the most common route is ODBC, Microsoft's older universal database API, or ADO, the newer COM-based equivalent.

The ODBC drivers for SQL Server usually work well, so do not be put off by bad experiences with drivers for Microsoft Access, Visual FoxPro, or other desktop drivers.

If you have Visual Basic 6.0, Visual InterDev or a compatible version of Delphi 5.0, ADO is usually a better choice.

Here are three tips for getting started with SQL Server programming.

First, whatever API you use, you need to learn Transact SQL, the SQL Server variant of Structured Query Language. There are differences, although Microsoft has published a list of these and has been trying to reduce them.

Second, to get the best from SQL Server you need to use stored procedures, which are SQL scripts that are compiled and stored with the database. Stored procedures can take parameters. Most Access queries, particularly those that take parameters, should be converted to stored procedures.

Third, consider adding timestamp fields to SQL Server tables. These fields speed up some internal processes and also make it easy to do optimistic locking. The timestamp indicates when the record last changed, so you can use it to detect if one user has updated the data while another user had the same record open for editing.

Mysteriously, Microsoft states that MSDE is tuned for use by five or fewer users

Microsoft has taken a few steps to protect sales of SQL Server licences. Individual databases are limited to 2GB. MSDE does not support transactions across several SQL Servers.

Replication with an SQL Server database requires a licence. More mysteriously, Microsoft states that MSDE is tuned for use by five or fewer concurrent users, although this is not a licensing restriction.

How much performance deteriorates after that point is unclear – if you have tried it out, please send an email explaining just how it went.

Overall, the restrictions are reasonable and MSDE is a great deal for Windows developers. There is a lot more to learn, but try not to be put off from making a start.

VB and Outlook

Alan Lhermette wrote in to ask how he could customise dialog boxes generated by clicking on the Full Name command button in a standard contact form. Despite reading a number of books on Outlook, VBA and VBScript he still can't seem to find the answer.

Outlook 2000 comes with two programming languages. The application itself includes VBA, just like the other Office applications, but VBA is not supported in Outlook forms. Forms only support VBScript.

There is no VBScript dialog editor, so on the face of it you are limited to InputBox, which throws up a single edit box. There are a couple of ways around this.

First, you can use one of the spare tabs on the contact form instead, although this is not quite a dialog.

Second, you can design the dialog in VBA and call it from VBScript. Open the VBA editor, insert a UserForm, and design the dialog there.

Next, create a sub-routine without any parameters that displays the UserForm modally.

Fig 1 shows an example. Now you can call the dialog from VBScript in a contact form, as Application.showdialog.

PCW CONTACTS

Tim Anderson welcomes your Visual Programming comments and queries. Contact him at visual@pcw.co.uk or via the PCW editorial office.

For more information about MSDE, see http://msdn.microsoft.com/vstudio/msde/techfaq.asp