Book 3

Database Connectivity

U:\Book\Book_03.doc

Database Connectivity

Fehler! Es wurden keine Einträge für das Inhaltsverzeichnis gefunden.

1 Database Access With Windows ODBC

Most web projects are database projects. Although we want to make use of R/3 as the main intelligent database server, we will need to store data additionally or alternatively in databases on the web server. The chapter deals with different database access methods, mainly through ODBC, JDBC and SQL. These techniques had been introduced to allow a common access gateway to database independent of the programming language and the database engine.

1.1 OLE/DB, ODBC and other Data Source Driver Models

Accessing a database with its native driver may give you additional programming capabilities while losing some compatibility. Many data sources do not even support ODBC.

The Open Database Connectivity ODBC has been the long-standing Windows standard solution for connecting your applications to arbitrary databases. The principle is to use virtually the same interface structure, no matter what physical kind of database is connected. However, it is certainly not the only one.

Connecting to your database with ODBC avoids the need for detailed knowledge of individual database languages for programmed access

ODBC is a universal database access interface. Its purpose is to allow a unified access method to connect to different databases. ODBC allows one single place in the system or program where you specify the type of database or driver to use. Every access to ODBC will then be translated by the ODBC driver to the individual database language, be it SQL, Access (JET), dBase, Excel or flat text files.

OLE/DB is Microsoft's latest proposal for database connectivity

Since Microsoft pushes the use of their MS SQL Server, they discourage the further use of ODBC. As a replacement they suggest using the OLE/DB driver model. OLE/DB features the same capabilities as ODBC plus the support of SQL. However, the driver model is totally different from ODBC, so using OLE/DB in place of ODBC would mean some rewriting of your applications.

DAO – Database Access Object uses the Access native JET engine

There is another commonly used access method mainly for MS Access databases, the Database Access Object Control. This is an Active/X (OCX) library which implements access methods for Microsoft's JET database engine, which is mainly used by Microsoft Access, but is also able to read and write dBase and FoxPro files. If you implement a purely Access based application, you may like it because of the wealth of features of the DAO, especially the possibility to fully access the data dictionary of an Access database.

Kommentar: Is this repetition with 2 lines above..?

Use IBM Database V or SQL server for large volume database

However, if you expect large data volumes for your local web server database then you should consider MS SQL Server or IBM Universal Database and then DAO is out of the reckoning.

SAP R/3 is connected via DCOM object methods or simulated RFC calls

There is neither a commonly used ODBC nor OLE/DB driver for SAP R/3 available to date, although you could easily implement such an ODBC or OLE/DB driver for R/3 by mapping RFC calls to SQL stored procedure calls and SQL statements to dynamic SQL statements. Therefore we will access R/3 data sources with a native DCOM connection by either executing a BAPI object method or by calling a function module via RFC. It might be interesting to know, for the techies only, that RFCs are usually called via DCOM by calling a DCOM Proxy method which tells R/3 to call the right function.

Figure 1: Connection schemes for different access methods

Scheme here (from msdn???? Or Addison Wesley VB for DB programmierung???)

1.2 Setting up A Sample Database

In this chapter we will show how to work on one of the sample databases found in Microsoft Access. The database used is created from the “Order Entry.mdz” of MS Access 97.

Microsoft Access comes with a number of template databases which have the extension .mdz

Microsoft Access comes with a couple of sample databases. Of course you may define your own databases with the Access designer, but making use of the sample databases is a good starting point, especially because the databases are already filled with sample data. This is great because there is nothing more annoying for a developer of a database application than an empty or nearly empty database. The templates which you can use to generate the sample databases are stored in the Microsoft Access template folder which is automatically installed with Access. The templates have the ending `.mdz` so it is easy to find them. In our installation we found them in the standard folder

```
C:\Program Files\Microsoft Office\Templates\Databases.
```

In Access 2000 there is the sample database northwind.mdb

We decided to use a template from MS Access 97 as we learned from experience that there is still a lot of companies out there that migrated to Microsoft Office 2000 but not to Access 2000. There is a good reason for it. While Word 2000 and Excel 2000 are compatible with their predecessors, e.g. you can save in Word 97 format with Word 2000, this is not true for MS Access 2000. MS Access 2000 can open MS Access 97 databases, but you cannot modify data dictionary objects like adding or modifying tables or queries and save them directly in MS 97 format (however, it is possible to convert MS 2000 databases into MS 97 format). For Microsoft Access 2000 users there is the standard example database Northwind, which you can also download from http://msdn.microsoft.com .

We created a new database weborders.mdb from the template Order Entry.mdz

For our first trials we create a new database with the name weborders from the template “Order Entry.mdz”. This is a sample application for sales order entry in Microsoft Access. Initially we are only interested in the “`Order Details`“ database table.

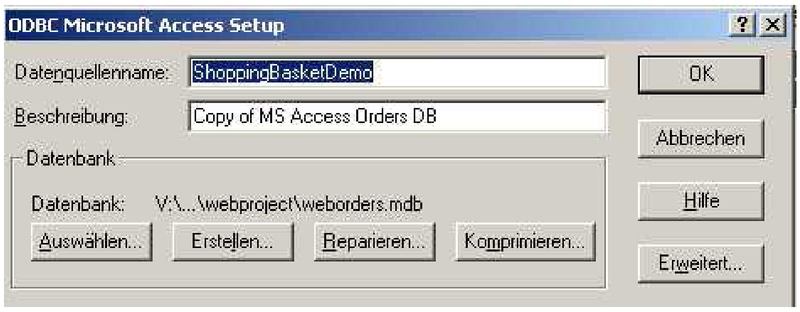
In the next step we will register the database as an ODBC source

In the following chapters we will refer to the database as an ODBC data source with the name ShoppingBasketDemo . This name must be registered and assigned to the database in the windows control panel. To register do:

- Go to the Control Panel and choose ODBC Sources
- Choose the System DSN tab strip
- Enter the path name to your database weborders.mdb and specify the name ShoppingBasketDemo

This will make your database known to all ODBC compliant objects.

Figure 2: Datasource (ODBC) screen in the Windows Control Panel


ODBC Microsoft Access Setup

Datenquellenname: ShoppingBasketDemo

Beschreibung: Copy of MS Access Orders DB

Datenbank

Datenbank: V:\...\webproject\weborders.mdb

If you want to access the data from the newly copied or created database you can connect via ODBC

The actual driver to be used by ODBC is either specified during connection to the database in the *connection string* or in the central configuration folder of the ODBC configuration as a so called *DSN* file. The DSN holds a short name for the complete connection and driver information that may be required by the accessed database.

Figure 3: Connect to a database by specifying the driver directly

```
conn.Open _
"Driver={Microsoft Access Driver (*.mdb)};DBQ=" _
 & Server.MapPath("../../dbs/orders.mdb")
conn.Close
```

Figure 4: Connect to a database by using a DSN name previously defined in the ODBC configuration

```
conn.Open "DSN=weborders"
conn.Close
```

1.3 Selecting Data From Database Tables

The Microsoft VB ADO object makes it very simple to read, insert, update and delete data from a database table.

Microsoft ADODB and RECORDSET are powerful tools when working with databases

The next example makes use of the ADODB and RECORDSET objects. The ADODB object exposes the basic methods to access a database table. A recordset is an enhanced version of an array. While you can access arrays only via indices, a recordset comes with database-like access methods.

We create a new database weborders.mdb from the template Order Entry.mdz

For our first trials we create a new database with the name 'weborders' from the template Order Entry.mdz. This is a sample application for sales order entry in Microsoft Access. Initially we are only interested in the Order Details database table.

Connecting to your database with ODBC avoids the need for detailed knowledge of individual database languages for access through a program

ODBC - Open Database Connectivity - is a universal database access interface. Its purpose is to provide a unified access method for connecting to different databases. ODBC allows one single place in the system or program where you specify the type of database or driver to use. Every access to ODBC will then be translated by the ODBC driver to the individual database language, be it SQL, Access, dBase, Excel or flat text files.

If you want to access the data from the newly copied or created database you can connect via ODBC

The actual driver to be used by ODBC is either specified during connection to the database in the connection string or in the central configuration folder of the ODBC configuration as a so-called DSN file. The DSN holds a short name for the complete connection and driver information that may be required by the accessed database.

Figure 5: Reading records from a database table and output them as an HTML response

```
<TABLE BORDER="1" WIDTH="100%">
<TR><TD>
<%
set conn = Server.CreateObject("ADODB.Connection")
set recs = Server.CreateObject("ADODB.Recordset")
conn.ConnectionString = _
"Driver={Microsoft Access Driver (*.mdb)};DBQ=" _
 & Server.MapPath("../../dbs/orders.mdb")
conn.Open
'Orders is the name of a table of the open database
recs.Open "Select * from Orders", conn
number_of_recs_to_display = 8
line_separator = "</TD><TD>"
field_separator = "</TR><TR><TD>"
response.write recs.GetString( , _
 number_of_recs_to_display, _
 line_separator, _
      field_separator, "")
conn.Close
%>
</TD></TR></TABLE>
```

recordset.GetString returns all field values of a number of rows as a single string separated by specified separator Examples

The example makes use of the very powerful GETSTRING method. It relieves the programmer of a lot of work. It loops over the recordset and concatenates all the field contents. Fields are separated by the field separator and lines are separated by the specified line separator. We chose separators which build an HTML table.

openodbc_dsn.asp Open ODBC source via a DSN previously registered in the control panel

openodbc_dbq.asp Open an ODBC source directly by specifying a full connection string

130

1.4 Selecting Data Using DAO

The Microsoft VB DAO "Data Access Object" is another way to access Microsoft databases, mainly those driven by the Microsoft JET engine like Microsoft Access. DAO is less efficient than ADO but gives you access to some features of ACCESS database not accessible via SQL.

Microsoft DAO and RECORDSET are powerful elements to work with databases

The next example makes use of the Microsoft DAO to access the data dictionary to determine the names of all tables in the database and to explore its content. For that purpose it does:

135

- Open a database as an Microsoft DAO
- Read the names of all tables in the data dictionary
- Display some records of every table found

Figure 6: Loop over all tables in a JET database and list their contents completely

```
<%
Sub ShowRecs(pName)
if left(pName,4) = "MSys" then exit sub
Set recs = dbs.OpenRecordset(pName)

response.write "<HR></HR>"
response.write "<H3>Contents of table:&nbsp;<em>" _
                     & TableDef.Name & "</em></H3>"
response.write "<TABLE BORDER='1' WIDTH='100%'> <TR>"

for each xfield in recs.Fields
  response.write "<TH>" & xfield.Name & "</TH>"
next

response.write "</TR>"

'*** Loop over all records in the set until EOF
while not recs.EOF
  response.write "<TR>"
'      *** Loop over the FIELDS collection
  for each xfield in recs.Fields
      response.write "<TD>" & xfield & "</TD>"
  next
  Response.Write "</TR>"
  recs.MoveNext
wend
response.write "</TABLE>"

End Sub
' ==============================================
Set dbengine = CreateObject("DAO.DBEngine.36")
Set dbs =
dbengine.OpenDatabase(Server.MapPath("..\..\dbs\orders.mdb"))
' ==============================================
for each TableDef in dbs.TableDefs
ShowRecs(TableDef.Name)
next
' ==============================================
dbs.Close
%>
```

Examples

openodbc_dsn.asp access a Microsoft Access database with a DSN string

openodbc_dbq.asp standalone access to a Microsoft Access database

1.5 ADODB Recordsets

Recordsets are the Visual Basic version of ABAP internal tables. They are structured arrays of dynamic size.

ADODB are usually automatically typed when a database connection is opened

Originally the ADODB recordsets were designed as dynamic buffers for database retrievals. Every time a database connection is opened, the recordset is restructured to reflect the structure of the retrieved database table.

```
Dim myRecs As ADODB.Recordset
Sub aConnectedRecordset()
    Set myConn = CreateObject("ADODB.Connection")
    myConn.Open "Driver={Microsoft Access Driver (*.mdb)};DBQ=" _
    & "U:\dbs\weborders2000.mdb"
    myRecs.Open "Select * from Orders", myConn
    myRecs.Close
End Sub
```

ADODB.Recordset in Microsoft Active/X Data Objects Library (msado15.dll)

The ADODB library provides the recordset class along with all the other data access objects for OLE/DB.

```
Dim myRecs As ADOR.Recordset
Set myrecs = CreateObject("ADODB.Recordset")
```

ADOR.Recordset in Microsoft Active/X Data Object Recordset Library (msador15.dll)

There is a light version of the ADODB recordset library in the msador15.dll library and referenced as ADOR. It uses less memory and creation overhead and is especially useful if you use disconnected recordsets only.

```
Dim myRecs As ADOR.Recordset
Set myRecs = CreateObject()
```

Disconnected recordsets can be used without a database connection

Recordsets can also be created programmatically by specifying the fields and field structure information. After this has been done the recordset must be opened with an empty connection string in order to instantiate the recordset.

```
Dim myRecs As ADOR.Recordset
Sub aDisconnectedRecordset()
  Set myRecs = CreateObject("ADOR.Recordset")
  myRecs.CursorLocation = adUseClient
  myRecs.Fields.Append "Name", adVarChar, 30, adFldIsNullable
  myRecs.Fields.Append "City", adVarChar, 30, adFldIsNullable
  myRecs.Fields.Append "CreditLimit", adNumeric, 10, adFldLong
  myRecs.Open
  myRecs.AddNew
  myRecs.Fields("Name") = "Micky"
  myRecs.Fields("City") = "Ducktown"
End Sub
```

In ABAP this can be achieved by defining an internal table:

```
DATA: BEGIN OF myitab OCCURS 0,
  Name(30), City(30), CreditLimit TYPE P,
END OF myitab.
```

Pseudo-disconnected recordsets can be created by defining the structure as a database table and leaving the table empty

For a bigger database project it is convenient to create the recordset with a reference to a data dictionary structure. I recommend that you define a table in a local database and leave this table without data. Then you can open the table and have the recordset typed to the structure. In ABAP you would define a structure or table with SE11 and use that as a reference like this:

```
  DATA: myitab LIKE sflight OCCURS 0 WITH HEADER LINE.
```

This takes table SFLIGHT as template for the fields of the internal table. The OCCURS parameter is mandatory and gives an estimate, how many records can be expected, a zero leaves the estimation to the ABAP engine. The addition WITH HEADER LINE automatically defines a buffer line to hold the data of a single record.

Looping over record set entries

The individual records of a recordset can be accessed by moving the access pointer forward and backward using the MoveFirst, MoveNext, MovePrevious or MoveLast methods.

Dim myRecs As ADODB.Recordset

```
Sub aConnectedRecordset()
    Set myConn = CreateObject("ADODB.Connection")
    myConn.Open "Driver={Microsoft Access Driver (*.mdb)};DBQ=" _
    & "U:\dbs\weborders2000.mdb"
    myRecs.Open "Select * from Orders", myConn
    myRecs.MoveFirst
    While Not myRecs.EOF
'      This loops over each field in the recordset
      For Each xfield In myRecs.Fields
                Debug.Print xfield
      Next
      myRecs.MoveNext
 Wend
    myRecs.Close
End Sub
Individual field can be accessed via its name:
 debug.Print myRecs.Fielöds("Name")
In ABAP this is achieved with the LOOP AT .. ENDLOOP statement.
LOOP AT myitab.
 Write: / myitab.name, myitab.city, myitab.CreditLimit
ENDLOOP.
Not naming the fields explicitly works in ABAP as well.
DATA: <ptr>.
LOOP AT myitab.
 WRITE: /.  "New line only
 DO 3 times.
      ASSIGN FIELD sy-index OF STRUCTURE myitab TO <ptr>.
      WRITE: <ptr>.
 ENDDO.
ENDLOOP.
```

1.6 VB Example: Display Order Details from the Northwind Database

The Northwind database comes ready and filled with data with the Microsoft Access installation.

Provided that you copied the Northwind.mdb database somewhere and registered it as ODBC source with the name Northwind the following example will list all the data of the [Order Details] table of the Northwind database.

```
Global conn As ADODB.Connection
Global recs As ADODB.Recordset

Sub Main()
    Set conn = CreateObject("ADODB.Connection")
    conn.ConnectionString = "DSN=Northwind"
    Set recs = CreateObject("ADODB.Recordset")
    If conn.State = 0 Then conn.Open
    recs.Open "Select * from [Order Details]", conn
    Call DataDisplay
    If conn.State = 1 Then conn.Close
End Sub

Sub DataDisplay()
    Dim xRow
    Dim xField
    For Each xField In recs.Fields
       Debug.Print xField.Name,
```

2 Accessing Databases With Java JDBC

What ODBC is for Windows, that is JDBC for Java. JDBC stands for Java Data Base Connectivity and is a neutral gateway interface allowing to use the same syntax for submitting database statements regardless of the underlying physical database implementation.

Accessing Databases With Java JDBC

What ODBC is for Windows, that is JDBC for Java. JDBC stands for Java Data Base Connectivity and is a neutral gateway interface allowing to use the same syntax for submitting database statements regardless of the underlying physical database implementation. 185

2.1 JDBC – An ODBC For Java

3 Accessing Databases With R/3

R/3 does not natively support any database bridges like ODBC or JDBC. Instead it allows access to the underlying database through EXEC SQL by means of the Open SQL standard.

3.1 R/3 And Open SQL

An R/3 installation is an application package that is installed in top of a database instance. R/3 uses exclusively an Open SQL gateway to communicate with the database. Nearly every important database action, as record creation, modification and deletion can be achieved using regular ABAP IV statements.

In order to manipulate repository objects, like creating, modifying or deleting a database table structure, there is a complete set of RFC enabled function modules available in R/3. These function modules start with the prefix RPY….

If for some reason the ABAP functionality is not sufficient or unsatisfactory, ABAP permits the execution of “pass-through-SQL-statements”. In that case a valid SQL statement for the underlying database instance must be specified as a string and is executed through an EXEC SQL statement.