

Migrating a Microsoft Access Application into a .NET Solution that Uses SQL Anywhere

by Berndt Hamboeck
EASAC, SCAPC8 and CSI



www.iAnywhere.com

TABLE OF CONTENTS

THE BUSINESS PROBLEM	3
FROM ACCESS TO SQL ANYWHERE	3
REDESIGN OF THE DATABASE	6
THE PROTOTYPE.....	9
The DataWindow.Net ancestor object.....	9
Reading data from SQL Anywhere using OLEDB	12
CONCLUSION	13
About Berndt Hamboeck	13
ABOUT IANYWHERE	14

The Business problem

A new customer came with an existing application based on Microsoft Access database technology, and asked us to build a more robust application within a short timeframe. The customer wanted to address the limitations of the existing solution:

- Weak security
- Lack of storage configuration
- Lack of stored procedures, which they wanted to use to improve the application's reporting capabilities
- Chronic requirement to compact and rebuild the database
- Slow transaction processing and weak, limited indexing, which greatly limited reporting capabilities
- Support for only one connection to the database at a time, which did not allow it to scale to multiple users

After looking at the customer's requirements and a small demo at their site, we chose to convert the Access database to a SQL Anywhere database from iAnywhere. We decided to write the user interface and business logic using Microsoft's .NET platform. This required three steps:

- **Migrating the Access database to SQL Anywhere** – The Microsoft Access Migration Utility for SQL Anywhere made this the easiest part of the project. Within minutes we had a new database, including the complete schema and data. The old Access database was still available to verify that the new application was going in the right direction and no existing business logic was lost during the development cycle.
- **Modifying the underlying data schema to improve performance** – This was the most challenging and time consuming part of the project, as we had to go back several times during the definition phase to add new features requested by the customer. We used PowerDesigner Physical Architect, which is part of the SQL Anywhere installation, to modify the schema. It helped us keep existing data during the conversion, and enabled us to change the design without working directly on the database, which made it possible to try different database schemas. Last, but not least, it was possible to completely document the database for the customer, as the customer wanted to build their own new reports into the system.
- **Building the application using Visual Studio .NET 2003 and DataWindow.NET** – We chose Sybase's DataWindow.Net, because it is possible to create professional visual presentations from the database schema very quickly. This allowed us to rapidly iterate through several prototypes and illicit the customer's feedback. After the definition of the database and the design of the application's user interface, we implemented and tested the complete application. The last step was to localize the application into several different languages (at the moment the application can run in English, German, and Czech).

From Access to SQL Anywhere

While the schema of the Access database was generally going to stay the same, the database lacked primary keys, and we needed to convert the German table and column names to English since the application would be distributed to different countries where English is more universally known. It also included a lot of sample data that we wanted to keep in order to save time when it came to testing the new application.

We decided to use the Microsoft Access Migration Utility for SQL Anywhere in order to quickly import the database schema and sample data into SQL Anywhere. The free utility can be downloaded from the iAnywhere developer website at the following location:

http://www.iAnywhere.com/developer/code_samples/sqlany_migration_utility.html

The utility is a Microsoft Access add-in that migrates Microsoft JET technology database schemas and data into a SQL Anywhere database.


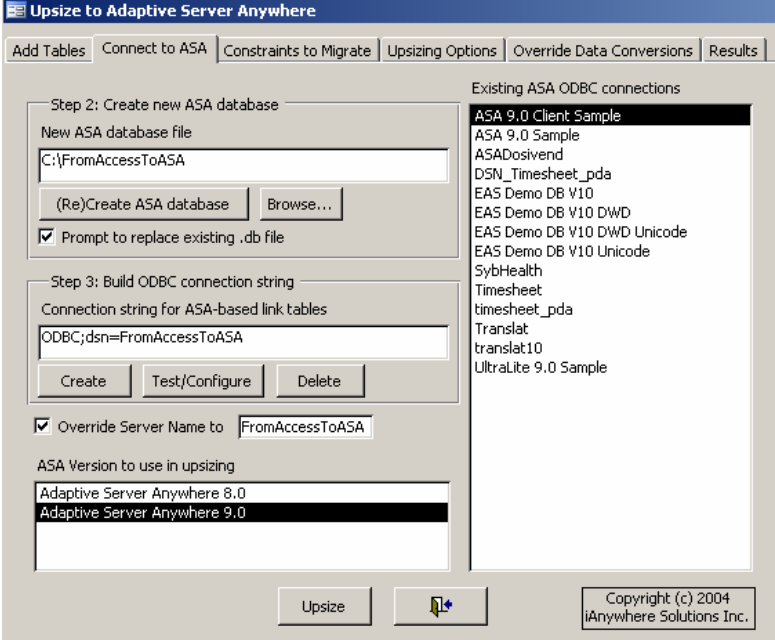
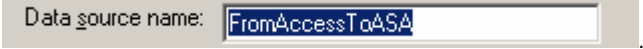
With the Access migration tool for SQL Anywhere, we could choose to continue using the Microsoft Access user interface (which we chose to replace later). The SQL Anywhere database engine simply replaces the Microsoft


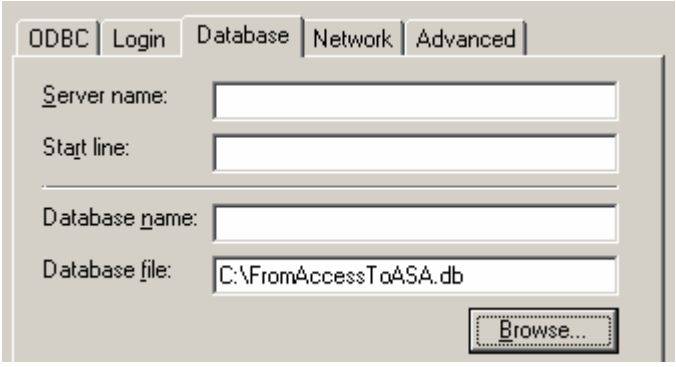
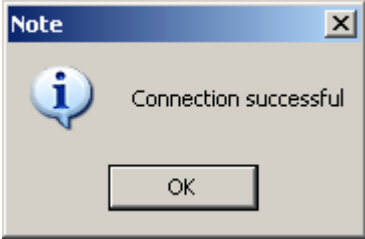
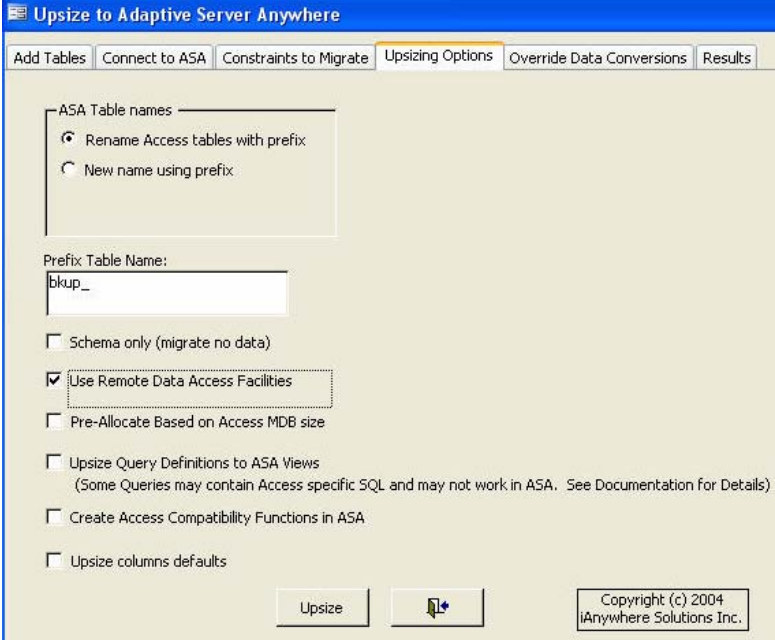
Migrating a Microsoft Access Application into a .NET Solution that Uses SQL Anywhere

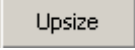
Berndt Hamböck

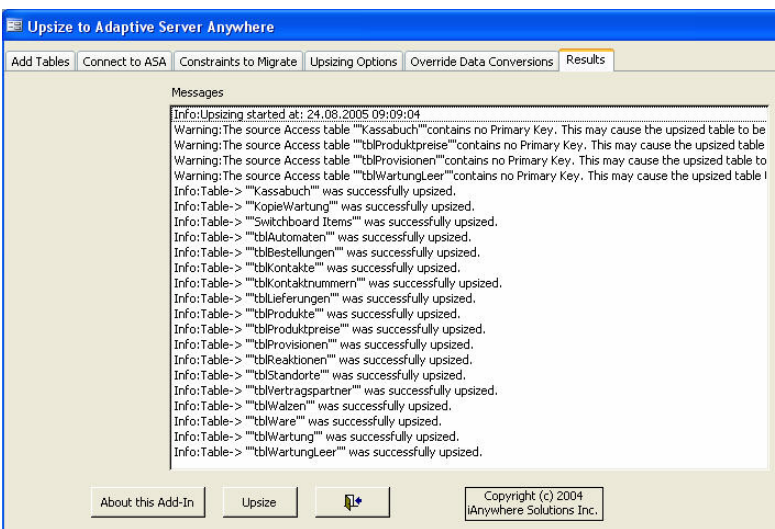
JET engine working behind the scenes after the data is migrated. So by using this tool we had a brand new SQL Anywhere database, including the existing data, within minutes.

Using the SQL Anywhere Migration Utility

	Action	Result
1	<p>First, backup your original Access database. As a result of the migration it will point to the newly created SQL Anywhere database (and you might still want to use the original application during the development process to look up business rules).</p> <p>Download migrationutility.zip and unzip it into a directory of your choice.</p> <p>Simply click on the extracted file ASAUpsizeAddin.mda.</p>	<ul style="list-style-type: none"> A backup of the original MS Access application. The upsize tool ASAUpsizeAddin.mda integrated into MS Access  <p>Note: Do NOT remove the file ASAUpsizeAddin.mda afterwards as this still needs to be available for MS Access to work correctly.</p>
2	<p>Open the existing application within MS Access.</p> <p>On the Add Tables tab choose all of the tables to upsize.</p> <p>On the Connect to ASA tab fill in the</p> <ul style="list-style-type: none"> New ASA database file name The ODBC connection string name The ASA version you are using 	
3	<p>Press the button (Re)Create ASA database</p>	<p>An empty SQL Anywhere Database file (c:\FromAccessToASA.db) is created.</p>
4	<p>Press the button Test/Configure</p>	<p>The ODBC administrator opens and the connection name is already filled in.</p> 

<p>5</p> <p>Fill in the values on the Login tab:</p> <ul style="list-style-type: none"> • User name: dba • Password: sql 	
<p>6</p> <p>On the Database tab fill in the values for the Database file:</p>	
<p>7</p> <p>Go back to the ODBC tab and press the button Test Connection</p> <p>Press the OK button to save the ODBC connection.</p>	<p>An ODBC data source is successfully tested and created.</p> 
<p>8</p> <p>Back within the upsize tool in MS Access, on the Upsizing Options tab check 'Use Remote Data Access Facilities'.</p> <p>Note: This option provides more efficient data movement especially when international numeric formats are involved, such as float, decimal (.), and currency that are formatted with characters other than decimal (.).</p>	

9 Now we are ready to upsize, press the button 



The wizard will switch automatically to the "Results" tab.
Now we have a new SQL Anywhere database filled with our tables and data.

Redesign of the database

We used Sybase PowerDesigner's Physical Architect (included in the Studio package of SQL Anywhere) to reverse engineer the database and have a detailed look at the existing design. The next step was to archive the model, which stores all model information. This makes it easy to make changes to the database and enables PowerDesigner to make the modifications to the database for you without losing any data (well, this is not absolutely true as you will see later, one has to take care during the modification process). To archive the model simply select the File->Save As menu, then select Archived PDM (XML) in the Save As Type dropdown listbox, and click Save.

The first thing we saw was that the model was using German table and column names, it was poorly designed, and some of the tables did not even have primary keys. So, there was indeed some redesign necessary. As an example I will show you what we did using the contract partner table. You see the original contract partner table in Figure 1.

tblVertragspartner		
<u>VP_ID</u>	integer	<pk>
Anrede	char(50)	
Vorname	char(50)	
Nachname	char(50)	
Firma	char(50)	
Telefon	char(20)	
Kontonummer	char(50)	
BLZ	char(6)	
Institut	char(50)	
UID-Nummer	char(50)	
VSt	smallint	

Figure 1: Original table

The table should contain all the necessary data about a customer, and it is needed to keep a history of the data. So, we started to change (and add) the table and column names as you can see Figure 2. To be able to provide a history, we modified the primary key to contain the date when this data is going to be effective (effectivedate).

cpartner		
<u>pid</u>	integer	<pk>
<u>effectivedate</u>	date	<pk>
title	char(50)	
firstname	char(50)	
lastname	char(50)	
company	char(50)	
phone	char(20)	
bankaccount	char(50)	
bankcode	char(15)	
uid	char(50)	
vat	bit	
zip	char(10)	
city	char(40)	
country	char(50)	
mobile	char(20)	

Figure 2: Modified table

After all modifications were done on this table, we wanted to apply these changes to the database. This is just one click in Sybase PowerDesigner's Physical Architect when the model has been archived. To do this select Database->Generate Database, click the Options tab, then select the Automatic Archive check box in the After Generation group box, and click OK. To apply the changes, we used Database->Modify Database, and after clicking OK. You may get the message seen in Figure 3 if your table contains data, as we did. You should not ignore this message as you would lose all of the test data within your table.

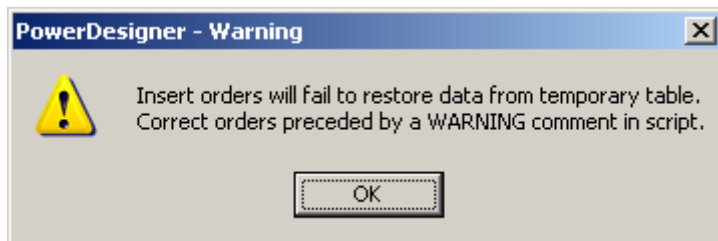


Figure 3: Warning during modify database

After accepting the message above, we had a closer look at the script generated by PowerDesigner (see Table 1). The section that needs our attention is this one:

```
--WARNING: The following insert order will fail because it cannot give value to mandatory columns
insert into cpartner (pid, effectivedate, title, firstname, lastname, company, phone, bankaccount, bankcode, uid, vat)
select VP_ID, ?, Anrede, Vorname, Nachname, Firma, Telefon, Kontonummer, BLZ, "UID-Nummer", VSt
from tmp_tblVertragspartner;
```

This statement would fail if executed, because the column effectivedate is not filled in by PowerDesigner. If you accidentally execute the statement, you will see that the original table is gone, but you will still have the data in a temporary table as PowerDesigner creates this one for us using the statement below:

```
alter table tblVertragspartner rename tmp_tblVertragspartner;
```

Take care to correct the above statement in a way that fills in all of the columns. In our example we use the getdate(*) function, as we do not have a history in the database, and all the contract partners are valid from today on for our development database. For the customer-partners table, the corrected statement would look like this:

```
insert into cpartner (pid, effectivedate, title, firstname, lastname, company, phone, bankaccount, bankcode, uid, vat)
select VP_ID, getdate(*), Anrede, Vorname, Nachname, Firma, Telefon, Kontonummer, BLZ, UID_Nummer, VSt
from tmp_tblVertragspartner;
```

```

/*=====*/
/* Database name: Database */
/* DBMS name: Sybase AS Anywhere 9 */
/* Created on: 13.06.2005 13:57:55 */
/*=====*/

alter table tblVertragspartner
delete primary key;

if exists(
select 1 from sys.systable
where table_name='tmp_tblVertragspartner'
and table_type in ('BASE', 'GBL TEMP')
) then
drop table tmp_tblVertragspartner
end if;

alter table tblVertragspartner rename tmp_tblVertragspartner;

/*=====*/
/* Table: cpartner */
/*=====*/
create table cpartner
(
pid integer not null default autoincrement,
effectivedate date not null,
title char(50),
firstname char(50),
lastname char(50),
company char(50),
phone char(20),
bankaccount char(50),
bankcode char(15),
uid char(50),
vat bit default Yes,
zip char(10),
city char(40),
country char(50),
mobile char(20),
constraint PK_CPARTNER primary key (pid, effectivedate)
);

--WARNING: The following insert order will fail because it cannot give value to mandatory columns
insert into cpartner (pid, effectivedate, title, firstname, lastname, company, phone, bankaccount, bankcode, uid, vat)
select VP_ID, ?, Anrede, Vorname, Nachname, Firma, Telefon, Kontonummer, BLZ, "UID-Nummer", VSt
from tmp_tblVertragspartner;

/*=====*/
/* Index: VP_ID */
/*=====*/
create unique index VP_ID on cpartner (
pid ASC,
effectivedate ASC
);

/*=====*/
/* Index: UID_Nummer */
/*=====*/
create index UID_Nummer on cpartner (
uid ASC
);

/*=====*/
/* Index: Kontonummer */
/*=====*/
create index Kontonummer on cpartner (
bankaccount ASC
);

```

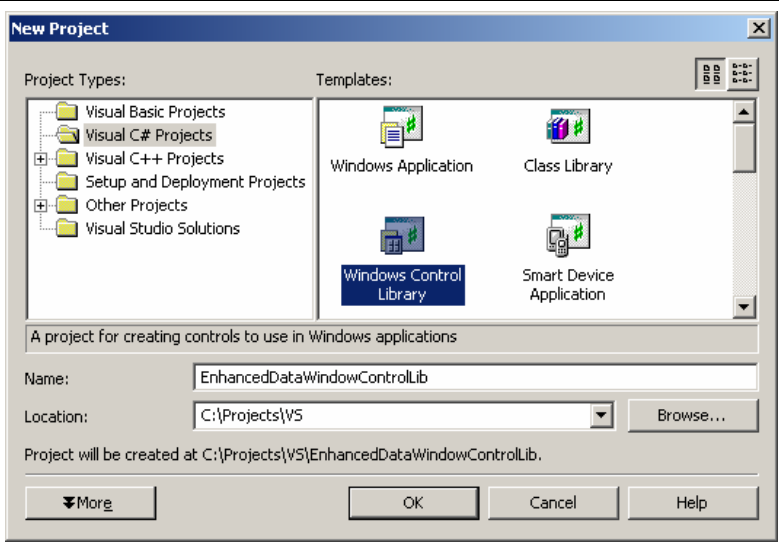
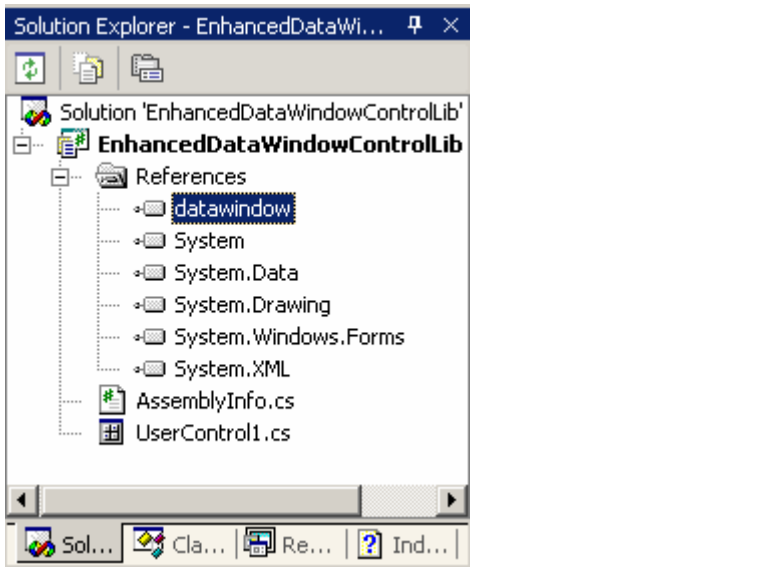
Table 1: Generated modify script

The Prototype

After redesigning the database, we were able to design the GUI fairly quickly by generating the needed DataWindows using Sybase's DataWindow.NET designer and placing them into a MDI application in Visual Studio 2003.

The DataWindow.Net ancestor object

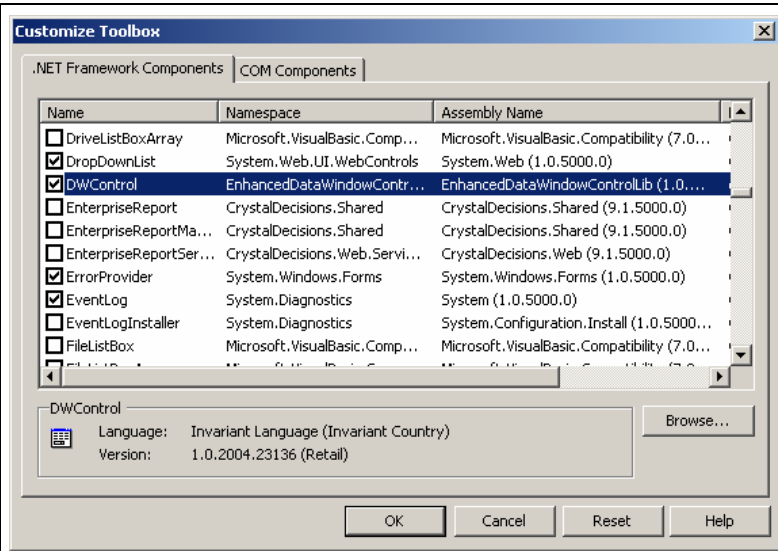
We extended the base DataWindow.NET control in order to customize its functionality. In this section, I will describe how this can be done easily. Our new DataWindow control will sort the data when a user clicks a header column.

Creating a DataWindow Ancestor	
Action	Result
<p>1 Open Visual Studio 2003 and on the File menu, point to New and then click Project to open the New Project dialog box.</p> <p>Select the Windows Control Library project template from the list of projects, and type EnhancedDataWindowControlLib in the Name box and select a location of your choice.</p>	
<p>3 In Solution Explorer, right-click References and select 'Add Reference' from the shortcut menu. Browse to your Datawindow.Net 1.5 installation directory and add datawindow.dll.</p>	

<p>4 In Solution Explorer, right-click UserControl1.cs and select View Code from the shortcut menu.</p> <p>Locate the Class statement, Public Class UserControl1, and change UserControl1 to DWControl to change the name of the component.</p> <p>Locate the statement Inherits System.Windows.Forms.UserControl, and change System.Windows.Forms.UserControl to Sybase.DataWindow.DataWindowControl.</p> <p>This allows the inherited control to inherit all the functionality of the DataWindow control.</p> <p>Add a bool variable that will indicate whether the DataWindow should sort when a user clicks on the header.</p> <p>In Solution Explorer, click UserControl1.vb and in the Properties window, change the FileName property to DWControl.cs.</p>	<pre>namespace EnhancedDataWindowControlLib { /// <summary> /// Enhanced DW Control /// </summary> public class DWControl : Sybase.DataWindow.DataWindowControl { /// <summary> /// Required designer variable. /// </summary> private System.ComponentModel.Container components = null; private bool sortOnClick = true; public bool SortOnClick { get { return sortOnClick; } set { sortOnClick = value; } } } }</pre>
<p>5 Now we need to implement the Click event so that if a user clicks in the header column of the DataWindow the data is sorted automatically.</p> <p>On the Build menu, click Build Solution</p> <p>From the File menu, choose Save All to save the project.</p> <p>Note: Controls are not stand-alone projects; they must be hosted in a container. In order to test the control, you must provide a test project within which it runs.</p>	<pre>protected override void OnClick(EventArgs e) { int iRow; string sColumn; base.OnClick(e); if (SortOnClick) { Sybase.DataWindow.ObjectAtPointer obj = this.ObjectUnderMouse; sColumn = obj.Gob.Name; //Only sort if user clicked into the header and on a column if (!(obj.Band.Type == Sybase.DataWindow.BandType.Header) sColumn == "DataWindow") return; string sCurrent = this.Describe("DataWindow.Table.Sort"); string sort = sColumn.Substring(0, sColumn.Length - 2); if (sCurrent.StartsWith(sort)) { if (sCurrent.ToUpper().IndexOf(" A") > 1) sort += " D"; else sort += " A"; } else sort += " A"; this.SetSort(sort); this.Sort(); } }</pre>

6 In the Toolbox, click My User Controls, right-click and select **Add/Remove Items...** from the shortcut menu.

Use the button **Browse...** to use the library we built before (EnhancedDataWindowControlLib).



7 Add a new Windows Application project to the solution, and name it EnhancedDWTTestApp.

In the Toolbox, click My User Controls.

Scroll down until the icon for DWCControl comes into view.

Add a copy of DWCControl to the form.

Set the properties of these controls as follows:

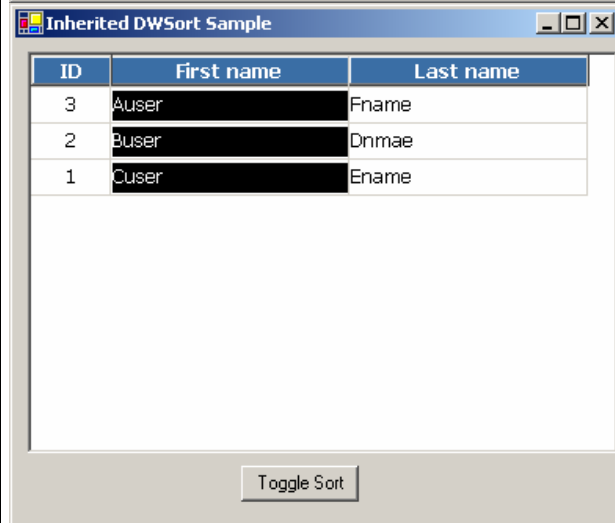
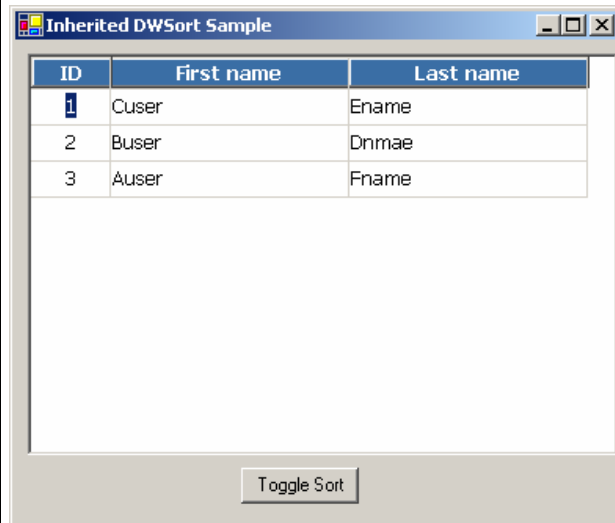
- Set the LibraryList
- Set a DataWindowObject

Note: For testing purposes I created an external DataWindow using 3 columns and used the import function to populate the DW.

In Solution Explorer, right-click EnhancedDWTTestApp and choose Set as StartUp Project from the shortcut menu.

From the Debug menu, choose Start and click on any header filed to sort the data.

Additionally place a button on the Form and code



<p>8 Additionally place a button on the Form and code it the way that it toggles the sort functionality on or off.</p>	<pre>private void button1_Click(object sender, System.EventArgs e) { dwControl1.SortOnClick = !dwControl1.SortOnClick; }</pre>
--	--

Reading data from SQL Anywhere using OLEDB

Before we can display data in a DataWindow control, we have to get the data that is stored in our SQL Anywhere database through a connection. In DataWindow.Net, this can either be through a Sybase.DataWindow.Transaction object or an AdoTransaction object. We chose the AdoTransaction object over the Transaction object because we are then able to share a connection with other database constructs (like DataSets) within an application.

We wrote a class called AppConnection, which uses the function DbConnect to connect to the SQL Anywhere database through the OleDb (Microsoft's low-level API for access to different data sources) driver provided by the SQL Anywhere installation. It should be noted that SQL Anywhere also has full native support for ADO.NET and ODBC as communication methods from .NET applications.

By using the AppConnection class there are only three lines of code needed to get data into your DataWindow:

1. AppConnection.DbConnect("dba","sql");
2. dw.SetTransaction(AppConnection.SQLCA);
3. dw.Retrieve()

Note, that the first line is needed only once within your application, then you are connected to the database as long as the application is running.

```
using System;
using Sybase.DataWindow;

namespace DLTracker
{
    /// <summary>
    /// Connect to the ASA database
    /// </summary>
    public class AppConnection
    {
        private static System.Data.OleDb.OleDbConnection oleDbConn;
        private static AdoTransaction SQLCA;

        public static Sybase.DataWindow.AdoTransaction SQLCA
        {
            get
            {
                return SQLCA;
            }
            set
            {
                SQLCA = value;
            }
        }

        public static System.Data.OleDb.OleDbConnection OleDbConn
        {
            get
            {
                return oleDbConn;
            }
            set
            {
```

```
        OleDbConn = value;
    }
}

public AppConnection()
{
}

public static bool DbConnect(string sUid, string sPwd)
{
    // OleDbConn
    try
    {
        OleDbConn = new System.Data.OleDb.OleDbConnection();
        OleDbConn.ConnectionString = "Data Source=ASADosivend;
                                   Provider=ASAProv.90;
                                   User ID=" + sUid + ";Password=" + sPwd + ";";

        OleDbConn.Open();

        SQLCA = new AdoTransaction(OleDbConn);
        SQLCA.BindConnection();
        return true;
    }
    catch (System.Data.OleDb.OleDbException e)
    {
        string errorMessages = "";

        for (int i=0; i < e.Errors.Count; i++)
        {
            errorMessages += "Index #" + i + "\n" +
                "Message: " + e.Errors[i].Message + "\n" +
                "NativeError: " + e.Errors[i].NativeError + "\n" +
                "Source: " + e.Errors[i].Source + "\n" +
                "SQLState: " + e.Errors[i].SQLState + "\n";
        }
        System.Windows.Forms.MessageBox.Show(errorMessages);
    }
    return false;
}
}
```

Table 2: Connecting to the SQL Anywhere DB

Conclusion

Using SQL Anywhere, PowerDesigner Physical Architect, and DataWindow.Net we were able to convert the existing Access application into a robust and stable multi-user application within a very short timeframe. We were able to convert most of the pre-existing data into the new database layout and provide a stable, secure, and reliable environment for our customer. We started with 23 tables in the original database and ended with 39 tables. We would not have been able to do this within the two-month project timeframe without SQL Anywhere, PowerDesigner Physical Architect, and DataWindow.Net.

About Berndt Hamboeck

Berndt Hamböck is a senior consultant for BHITCON (which offer broad training and product solutions) and a member of TeamSybase. He co-authored two PowerBuilder 9 books, and is a frequent writer for the PowerBuilder Developer Journal. He can be reached at Berndt.hamboeck@pocketpb.com.

About iAnywhere

iAnywhere is the worldwide market leader in mobile and embedded databases, mobile middleware and mobile and remote device management. More than 15,000 customers and 1,000 partners rely on the company's award-winning products, including SQL Anywhere® Studio and XcelleNet frontline solutions. In addition, its AvantGo® mobile Internet service has more than ten million registered subscribers. iAnywhere is a subsidiary of Sybase, Inc. (NYSE: SY). Visit www.iAnywhere.com for more information.

www.iAnywhere.com

North America

T 1-800-801-2069

1-519-883-6898

Europe, Middle East, Africa

+44 1628 597 100

iAnywhere Solutions, Inc.

Worldwide Headquarters

One Sybase Drive

Dublin, CA 94568-7902

U.S.A.

contact_us@iAnywhere.com

Asia Pacific

+852 2506 8700

Japan

+81 3 5210 6380

iAnywhere Solutions is a subsidiary of Sybase, Inc. Copyright © 2004 iAnywhere Solutions, Inc. All rights reserved. iAnywhere, SQL Anywhere, Adaptive Server, Sybase, the Sybase logo, SQL Remote, OpenClient, Sybase Central, Power Designer Physical Architect, Transact-SQL, UltraLite and PowerBuilder are trademarks of Sybase, Inc. or its subsidiaries. Java and JDBC are trademarks of Sun Microsystems, Inc. All other trademarks are properties of their respective owners. ® indicates registration in the United States of America.